UNIVERSITAT POLITÈCNICA DE CATALUNYA

Escola Politècnica Superior d'Enginyeria de Manresa

Enginyeria de Sistemes TIC

Music Scanner

Joan Bruch Serra

Manresa - Catalunya

Juny del 2018

Vull dedicar aquest treball a totes les persones que creuen en la música com una eina d'expressió comuna per tota la societat. A la meva família i amics pel suport rebut en tot moment, als companys i professors que he conegut durant la carrera, i a tots els alumnes i professors d'escoles de música. Que la música no pari mai!

Índex general

Índex de figures

Índex de taules

Índex d'Apèndixs

CAPÍTOL 1

Introducció

Em considero un gran amant de la música. He estudiat música des dels 7 anys, i actualment continuo actiu dins el món musical. Em faig meva la frase de l'escriptor francès Victor Hugo quan diu que "La música expressa el que no pot ser dit amb paraules i allò que és impossible romandre amb silenci".

És degut a aquest interès en la música, juntament amb la informàtica, el que m'ha donat la idea de dissenyar una aplicació que, a partir d'una imatge d'una partitura sigui capaç de reconèixer els elements musicals d'aquesta, amb la finalitat de reproduir la música escrita en format auditiu.

1.1. Objectiu General

Tal com he explicat a la introducció, l'objectiu general del meu Treball de Final de Grau és aconseguir una aplicació que reconegui la música escrita en una partitura i la reprodueixi en format audio.

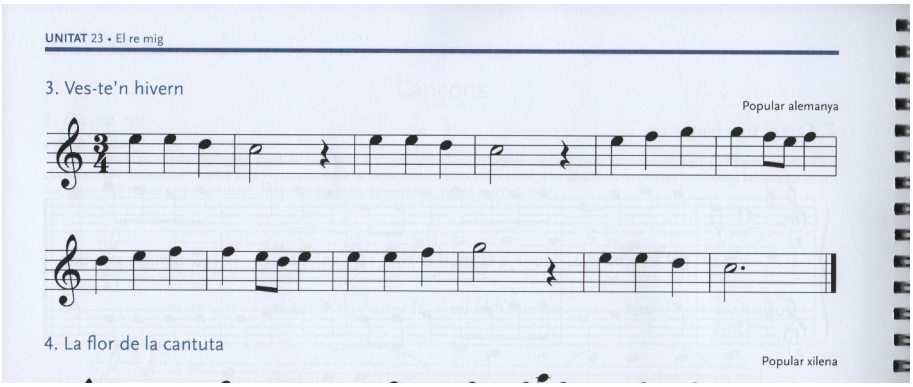


Figura 1.1: Exemple de partitura per processar. **Font:** Llibre "*Aprèn jugant amb la flauta travessera 1*" [1]

1.2. Objectius específics

Els objectius més específics són:

- Dissenyar, implementar i posar en marxa una aplicació de reconeixement de caràcters musicals, OMR [2]. S'ha optat per una programació orientada a objectes amb llenguatge *Python*, versió 3.0 o superior.
- Investigar i aprendre llibreries desconegudes fins ara per mi, de llenguatge *Python* com són OpenCV [3], Numpy [4] i MIDIUtil [5].
- Elaborar un estàndard amb el qual un alt tant per cent de les imatges que compleixin aquest estàndard seran reconegudes per l'aplicació Music Scanner.
- Crear una interfície web encapsulada des d'un servidor, amb la qual es pugui utilitzar l'aplicació Music Scanner.
- Crear una APP pel sistema operatiu Android, des d'on es pugui utilitzar l'aplicació Music Scanner.
- Investigar projectes semblants i comparar-los amb la meva aplicació.
- Posar a disposició d'un ampli entorn el resultat final de l'aplicació perquè tothom qui vulgui el pugui utilitzar i avaluar.

1.3. Requisits previs

L'aplicació ha de complir una sèrie de requisits que es detallen a continuació:

- Ha de ser robust i segur: no es pot posar en perill l'aparell des del qual s'està executant l'aplicació, ni s'han de posar en perill les dades de l'usuari que utilitza la pàgina web. L'aplicació també ha de reaccionar de forma adequada davant de situacions que, a priori, són imprevistes.

- Rapidesa en la seva execució. S'ha de mostrar el resultat final pocs segons després que l'usuari l'hagi executat, o avisar-lo que s'està executant amb una barra de progrés.
- Bona usabilitat de la interfície d'usuari: la interfície web ha de ser entenedora perquè tots els usuaris la puguin utilitzar correctament.

1.4. Estructura de la memòria

El treball consta de 8 capítols amb diferents apartats i subapartats. A continuació escric un breu resum de cada capítol per facilitar-ne la comprensió:

1. **Introducció:** En el primer capítol explico els motius pels quals vaig decidir realitzar un Treball de Final de Grau especialitzat en la música i programació, així com els objectius principals i els més específics.
2. **OMR: Estat de l'art:** En aquest apartat apareix informació sobre projectes semblants per conèixer com estan creats, quins mètodes, algorismes i procediments utilitzen per poder-ho aplicar a la meva aplicació, al mateix temps que mostro els principals programes d'edició i creació de partitures amb format digital.
3. **Organització del projecte:** Resumeixo les etapes del projecte. Com l'he subdividit en funció dels dies dedicats a les diferents tasques i el cost total de l'execució d'aquest.
4. **Descripció de l'aplicació:** Descric el projecte començant per la idea inicial, fins al resultat final. Explico l'arquitectura de l'aplicació, les llibreries utilitzades i els processos que es duen a terme dins de cada part del codi.
5. **Servidor web:** Comparo diverses opcions disponibles per crear un servidor, explico la opció que he triat i com l'he estructurat. També explico la seguretat de la pàgina web, una descripció i un manual d'usuari de la interfície.

6. **Resultats finals del programa:** Mostro els resultats i les conclusions dels tests que he realitzat amb diferents partitures.

7. **Conclusions:** Explico les conclusions finals del treball. Faig una llista dels objectius assolits.

8. **Línies futures:** Apareixen idees que m'han sorgit de possibilitats de millora, tant del sistema OMR com de la interfície web, i que no he pogut implementar a temps en el treball.

Nota: A l'apèndix A hi ha l'apartat anomenat *La Música: definicions i conceptes importants* escrit amb l'objectiu d'ajudar al lector a entendre els principals conceptes musicals que seran tractats en el conjunt del treball. Si tot i així hi ha conceptes que no s'entenen o no estan prou detallats, podeu consultar els tres primers volums dels llibres de *Teoria de la Música* [6] .

CAPÍTOL 2

OMR: Estat de l'art

Els sistemes de reconeixement s'utilitzen per realitzar tasques com la dels radars de trànsit per reconèixer les matrícules dels cotxes, la reproducció d'una cançó després d'escanejar la partitura, la digitalització de formularis escrits a mà alçada o de forma digital, el reconeixement facial, etc.

El meu cas concret, el reconeixement de caràcters musicals d'una partitura, és una tasca difícil de realitzar si es volen ocupar de tots els àmbits i elements existents a la música degut a la gran quantitat de símbols, notacions i combinacions entre aquests que poden aparèixer a qualsevol partitura. És per això que caldrà definir uns límits que formin part dels requisits estàndards de l'aplicació.

La tasca més important, i també la més complicada, dels sistemes OMR [2] és la de reconèixer partitures manuscrites. Molts compositors no van començar a escriure les seves partitures amb format digital fins la segona meitat del s.XX, coincidint amb l'aparició del programari de composició de música computada. Tant és així, que la digitalització s'ha utilitzat habitualment com una possible eina de conservació, oferint duplicacions fàcils, distribució i processament digital. La transcripció manual de les puntuacions musicals en un format digital adequat comporta molt de temps. El desenvolupament de mètodes generals de processament d'imatges per al reconeixement d'objectes ha contribuït al desenvolupament de diversos algorismes OMR. Aquests algorismes han estat fonamentals pel desenvolupament de sistemes

per reconèixer i codificar símbols de música per a una transformació directa de les partitures en un format simbòlic llegible per màquina.

Els objectius principals d'un sistema OMR són el reconeixement, la representació i l'emmagatzematge d'elements musicals en un format llegible per màquina. Un programa OMR hauria de poder reconèixer el contingut musical i fer l'anàlisi semàntic de cada símbol musical d'una partitura. Al final, tota la informació musical s'ha de guardar en un format de sortida fàcilment llegible per un ordinador.

Segons explica l'article [2] que tracta sobre OMR, un marc típic de reconeixement automàtic d'una partitura consta de quatre etapes principals:

1. Preprocessament d'imatges;
2. Reconeixement de símbols musicals;
3. Reconstrucció de la informació musical per construir una descripció lògica de la notació musical
4. Construcció d'un model de notació musical que es representarà com una descripció simbòlica del full musical.

En l'etapa de preprocessament d'imatges, es poden aplicar diverses tècniques, com ara la millora, la binarització, la rotació de la imatge,... per fer que el procés de reconeixement sigui més robust i eficaç.

La següent fase, reconeixement de símbols musicals, normalment es subdivideix en tres parts: (1) detecció i eliminació de les línies del pentagrama, per obtenir una imatge que conté només els símbols musicals; (2) segmentació primitiva de símbols; i (3) reconeixement de símbols.

La tercera i quarta etapa (reconstrucció de notació musical i construcció de representació final) poden estar intrínsecament entrellaçades. En l'etapa de reconstrucció de la notació musical s'utilitzen les regles gràfiques i sintàctiques musicals per introduir informació contextual per validar i resoldre ambigüitats. La sortida del sistema és un fitxer d'edició gràfica de música, com ara MIDI [7] MusicXML [8].

2.1. Aplicacions OMR disponibles

Actualment existeixen una gran varietat de productes i software capaços d'escanejar partitures, però l'origen el trobem l'any 1991, quan l'empresa *Musitek* [9] va llançar al mercat el primer software de reconeixement de partitures: *MIDISCAN* dissenyat pel sistema operatiu Windows. La línia de productes es va canviar posteriorment a *"SmartScore"* i es va tornar a llançar per a Windows 98 en 1998, i per Macintosh Power PC el 1999 com a producte híbrid de scanning / scoring. La versió actual, *SmartScore X2*, va ser llançada pel fabricant *Musitek* el 2013.

Cinc anys abans de l'última versió del *SmartScore*, l'any 2007, surt a la venda el programa *PhotoScore Ultimate 5* [10], el primer producte del món ideat pel reconeixement de música escrita a mà alçada i que també incloïa un sistema de reconeixement per la música impresa, convertint-se així en el primer programa amb dos opcions de reconeixement musical.

PhotoScore Ultimate 5 va ser el pioner en una multitud de programes de reconeixement que s'han anat desenvolupant amb el pas del temps. Actualment el seu rendiment (% de paraules escanejades) pot arribar fins al 99 % quan partitura escanejada està neta (sense deformacions, ni taques, ni talls i discontinuïtats a la imatge) i la notació que s'utilitza no conté símbols inusuals i poc utilitzats.

2.1.1. Aplicacions OMR Privades

A continuació s'expliquen un recull de programes OMR comercials que podem trobar disponibles a part dels ja comentats:

SharpEye

Actualment a la versió 2.68, el software *SharpEye* [11] només està disponible per sistema operatiu Windows. El resultat d'escanejar les partitures l'exporta a formats NIFF, MIDI i MusicXML (utilitzat per programes com *Finale* i *Sibelius*. Tot i que disposa d'un editor per corregir els errors de detecció, cal destacar que no funciona amb partitures escrites a mà alçada. La última versió disponible, 2.68 té un preu de 162,39€.

Capella

El software *Capella* [12] té la capacitat de reconèixer la partitura digital i exportar-la a format MusicXML. Amb la última versió, *Capella-Scan 8.0*, pots modificar els errors de detecció i també és capaç de reconèixer la lletra que hi hagi a la partitura. Disponible només per sistema operatiu Windows. Es pot adquirir per un preu de 199€.

PlayScore

PlayScore [13] és una aplicació OMR per a dispositius iOS i Android. Consta d'una versió Lite lliure de descàrrega i una altra de pagament. La versió de pagament permet reconèixer molts més caràcters musicals que la versió gratuita i a la botiga Google Play té un preu de 11,99€ . El resultat d'escanejar el transforma a format MusicXML o MIDI. Aquesta aplicació tampoc funciona amb partitures escrites a mà alçada, però ha investigant molt en la reproducció del resultat final, podent personalitzar els fragments de compassos que vols reproduir, el tempo, etc.

PDFtoMusic

Aquest programa, [14] permet extreure la partitura d'una imatge a partir de l'arxiu PDF que conté el document. Funciona amb Windows, Mac i Linux i el programa permet la reproducció i l'exportació a diferents formats (MIDI, WAV, MusicXML, etc). La versió completa té un preu de 199€.

2.1.2. Aplicacions OMR Lliures

També existeixen aplicacions de codi lliure per les quals no és necessària cap llicència pel seu ús. Algunes aplicacions són part de projectes o treballs d'investigació d'universitats o departaments d'investigació. A continuació faig esment de les dues aplicacions lliures més conegudes.

Audiveris

Audiveris [15] és un programa OMR de codi lliure amb l'última versió estable de data 4 d'agost del 2017 (versió 5.0). Està programat amb Java. El resultat d'escanejar les partitures l'exporta al format MusicXML. El codi font de l'aplicació el podem trobar a Github.

Gamera

Gamera és un altre programa OMR de codi lliure [16]. Aquest programa forma part d'un projecte de la Universitat Johns Hopkins desenvolupat per Karl MacMillan, Michael Droettboom i Ichiro Fujinaga. Tal com el defineixen a la seva pàgina web, no és un programa OMR empaquetat sinó un "kit"d'eines per la creació de sistemes de reconeixement d'imatges. Gamera és una biblioteca multi-plataforma escrita amb llenguatge Python. No és capaç de reconèixer partitures escrites a mà alçada i només carrega imatges amb els formats TIFF i PNG.

2.2. Algorismes de reconeixement de caràcters

En aquest apartat explicaré un seguit de mètodes i algorismes utilitzats en el reconeixement de caràcters musicals.

2.2.1. El llindar d'Otsu

El mètode d'Otsu [17] ,(en honor al seu inventor, Nabuyuki Otsu), és un mètode de binarització (convertir una imatge en escala de grisos a blanc i negre) que se sol utilitzar en el preprocessament de les imatges dels programes OMR.
Per transformar la imatge, aquest mètode utilitza tècniques estadístiques. En concret s'utilitza la variància (mesura de dispersió dels valors per mesurar la dispersió dels nivells de gris) per determinar el valor llindar de forma que la dispersió dins de cada segment sigui la més petita possible i al mateix temps la dispersió entre segments diferents sigui la més alta possible. És a dir, calcular les variàncies per cada nivell de gris i trobar el valor que la suma de les variàncies ponderades sigui mínim. A continuació explico l'aplicació del mètode Otsu:

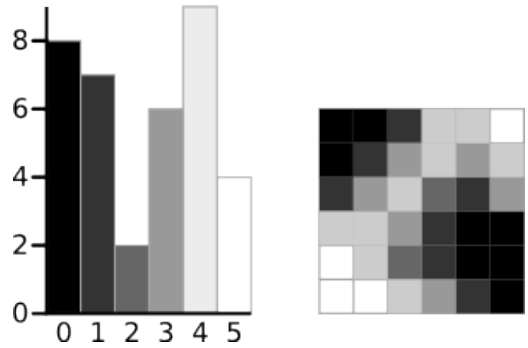


Figura 2.1: Imatge en escala de grisos de 6 nivells i el seu histograma
Font: labbookpages.co.uk/software/imgProc/otsuThreshold.html

A partir de la imatge en escala de grisos anterior, es vol transformar a blanc i negre (binarització). El primer pas és el de calcular la variància per cada nivell de gris.

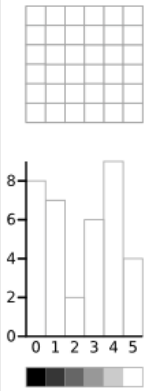
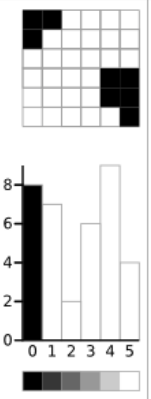
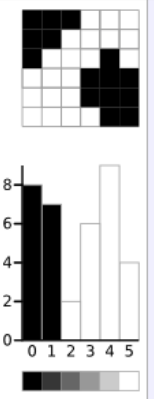
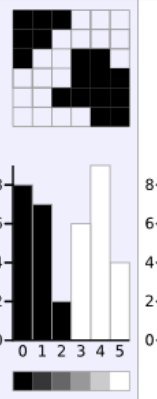
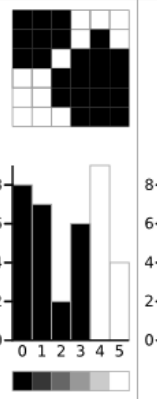
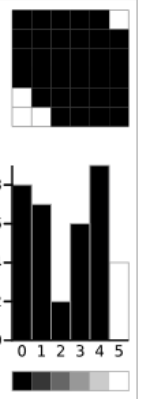
Threshold	T=0	T=1	T=2	T=3	T=4	T=5
						
Weight, Background	W_b = 0	W_b = 0.222	W_b = 0.4167	W_b = 0.4722	W_b = 0.6389	W_b = 0.8889
Mean, Background	M_b = 0	M_b = 0	M_b = 0.4667	M_b = 0.6471	M_b = 1.2609	M_b = 2.0313
Variance, Background	σ^2_b = 0	σ^2_b = 0	σ^2_b = 0.2489	σ^2_b = 0.4637	σ^2_b = 1.4102	σ^2_b = 2.5303
Weight, Foreground	W_f = 1	W_f = 0.7778	W_f = 0.5833	W_f = 0.5278	W_f = 0.3611	W_f = 0.1111
Mean, Foreground	M_f = 2.3611	M_f = 3.0357	M_f = 3.7143	M_f = 3.8947	M_f = 4.3077	M_f = 5.000
Variance, Foreground	σ^2_f = 3.1196	σ^2_f = 1.9639	σ^2_f = 0.7755	σ^2_f = 0.5152	σ^2_f = 0.2130	σ^2_f = 0
Within Class Variance	σ^2_W = 3.1196	σ^2_W = 1.5268	σ^2_W = 0.5561	**σ^2_W = 0.4909**	σ^2_W = 0.9779	σ^2_W = 2.2491

Figura 2.2: Càlculs corresponents per cada nivell de gris
Font: labbookpages.co.uk/software/imgProc/otsuThreshold.html

Podem comprovar que la columna de t = 3 és la que compleix la sentència que la suma de les variances ponderades és mínima. Per tant, el valor llindar Otsu per la imatge 2.1 és t = 3.

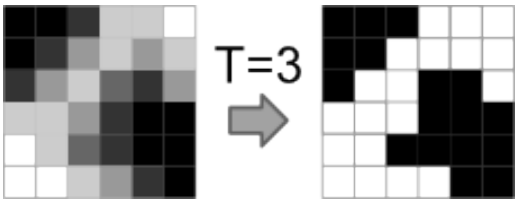


Figura 2.3: Resultat de transformar de grisos a blanc i negre amb l'algorisme Otsu
Font: labbookpages.co.uk/software/imgProc/otsuThreshold.html

Aquest mètode l'utilitzo a l'etapa del processament de les imatges per transformar les imatges de les partitures a blanc i negre.

2.2.2. kNN: k-Nearest Neighbors

L'algorisme kNN és un mètode de classificació d'objectes amb reconeixement de patrons [18]. Va ser publicat l'any 1951 pels investigadors E. Fix i J. Hodges. Es basa

en classificar un objecte segons la classe a la qual pertanyen els "k"veïns més propers. Per identificar els objectes, aquests es representen amb vectors p-dimensionals de característiques en un determinat espai multi-dimensional i es comparen amb els vectors dels veïns coneguts utilitzant la distància Euclidiana ,2.1, o altres mesures, com la distància Manhattan. Això permet dividir les regions de l'espai en diferents classes d' objectes.

$$d(P,Q) = \sqrt{\sum_{i=1}(p_i - q_i)^2} \text{ Equació de la distància Euclidiana} \qquad (2.1)$$

Poso un exemple per entendre'n l'algorisme: A la següent figura podem veure un exemple gràfic d'un espai amb dues classes diferents i hem de classificar l'element verd. Si k = 3 (primer cercle), classificarem l'exemple desconegut a la classe triangle, ja que només hi ha un quadrat i 2 triangles dins el cercle k=3. En canvi, si k=5, classificarem l'element desconegut a la classe quadrat ja que hi ha 2 triangles i 3 quadrats dins del cercle k=5.

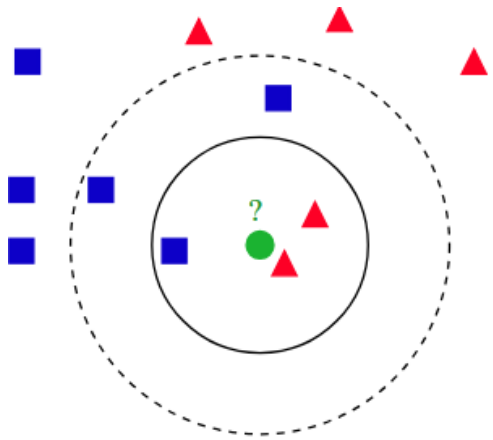


Figura 2.4: Exemple explicació algorisme kNN
Font: en.wikipedia.org/wiki/File:KnnClassification.svg

Segons aquest mètode, si existeixen gran quantitat d'elements irrellevants podem dur a terme una mala suposició de l'element desconegut. Per això és molt important extreure les característiques que representin de forma òptima les propietats de cada element i que permetin una fàcil diferenciació.

A l'hora de la implementació de l'algorisme cal tenir en compte l'elecció del valor *k*. Aquest depèn fonamentalment de les dades. Generalment valors alts de *k* redueixen l'efecte del soroll a la classificació però creen límits entre classes semblants. Un bon valor *k* pot ser escollit mitjançant una optimització del seu ús. El cas especial en què la classe està definida per ser la classe més propera a l'element desconegut és quan *k=1*, i s'anomena **Nearest Neighbor Algorithm** (algorisme del veí més proper).

Existeixen algunes variants de l'algorisme bàsic. Una de les més interessants consisteix a ponderar la contribució de cada veí d'acord a la distància entre ell i l'exemplar a ser classificat, donant més pes als veïns més propers.

2.2.3. Template Matching

Template matching (associació de patrons) [19], és una tècnica de processament d'imatges digitals que permet trobar petites peces d'una imatge que coincideixen amb una imatge d'una plantilla o patró. És una tècnica molt utilitzada en aplicacions de reconeixement de símbols i caràcters.

Aquesta tècnica compara tots els píxels de la imatge, per tant, per imatges grans, l'algorisme tardarà més temps a recórrer tota la imatge. S'aconsella si és possible reduir la quantitat de punts de mostreig reduint la resolució de les imatges i dels patrons pel mateix factor d'escala, realitzant l'operació de comparació a les imatges reduïdes resultants o definir una finestra de cerca de punts de dades dins de la imatge de cerca, de manera que la plantilla no hagi de cercar tots els punts de dades.

Aquesta tècnica la utilitzo per reconèixer els patrons a la partitura a través de la llibreria OpenCV, explicada al punt 4.3.1.

2.3. Programes de representació digital de partitures

Existeixen gran quantitat de programes pensats per la creació i edició de partitures en format digital. La gran majoria són utilitzats pels editors per crear i editar les seves peces però també es poden utilitzar aquests programes per donar a l'usuari una representació gràfica del resultat final de reconeixement OMR i facilitar-li la possibilitat d'editar ritmes, notes i altres paràmetres que no hagin estat ben reconeguts per l'aplicació OMR. A continuació explico breument els principals programes d'edició de partitures digitals.

2.3.1. MuseScore

MuseScore [20] és un programa de notació musical per a Linux, Mac OS X i Microsoft Windows. Es tracta d'un editor *WYSIWYG* (el que veus és el que aconseguiràs), amb suport complet per reproduir partitures i importar o exportar MusicXML i fitxers MIDI estàndard. Té suport per notació de percussió, així com impressió directa des del programa. MuseScore és programari lliure publicat sota la llicència Pública General de GNU.

El programa té una interfície d'usuari neta, amb una ràpida entrada de notes en edició similar a l'ingrés ràpid de notes que tenen altres programes comercials de notació musical, com *Finale* i *Sibelius*. És el programa que he utilitzat per editar les partitures. A l'apèndix C he escrit un manual pas a pas del funcionament.

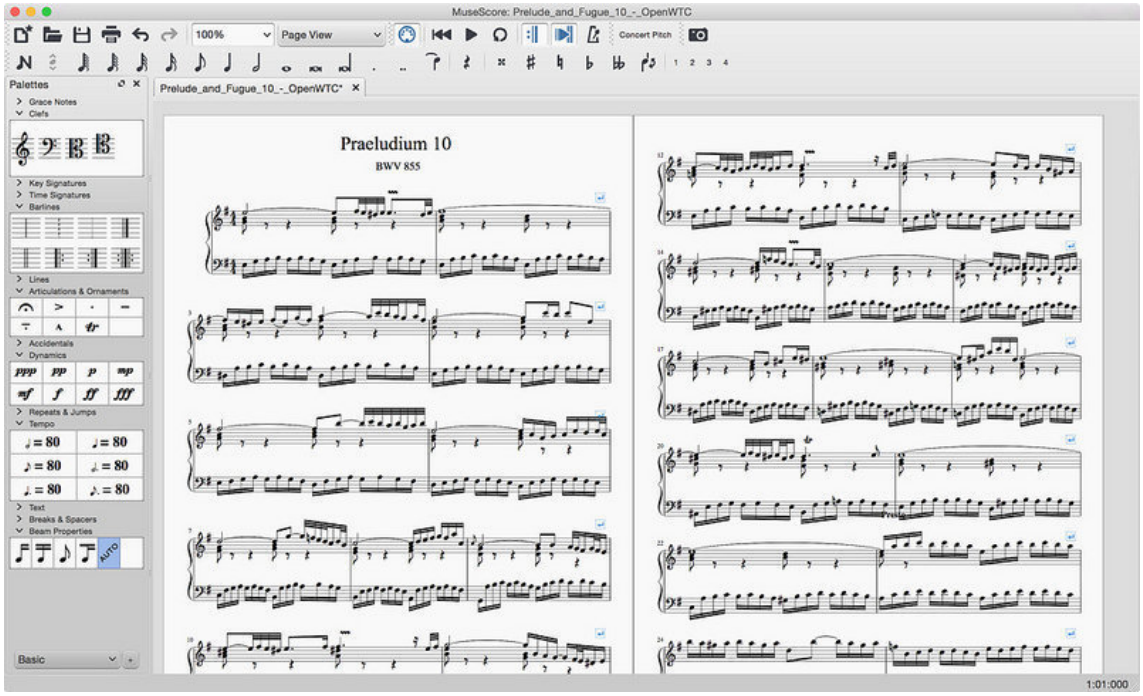


Figura 2.5: Captura de pantalla del programa MuseScore **Font:** topbestalternatives.com/wp-content/uploads/2016/09/MuseScore.jpg

2.3.2. Finale

Tot i ser un programa amb llicència de pagament, *Finale* [21] és l'editor de partitures més popular del mercat internacional per davant de *Sibelius*. Igual que *MuseScore*, es tracta d'un editor WYSIWYG i utilitza el protocol MIDI per reproduir les partitures. Un punt a favor respecte MuseScore és que Finale permet l'entrada de notes a temps real mentre toques un teclat MIDI connectat al programa.

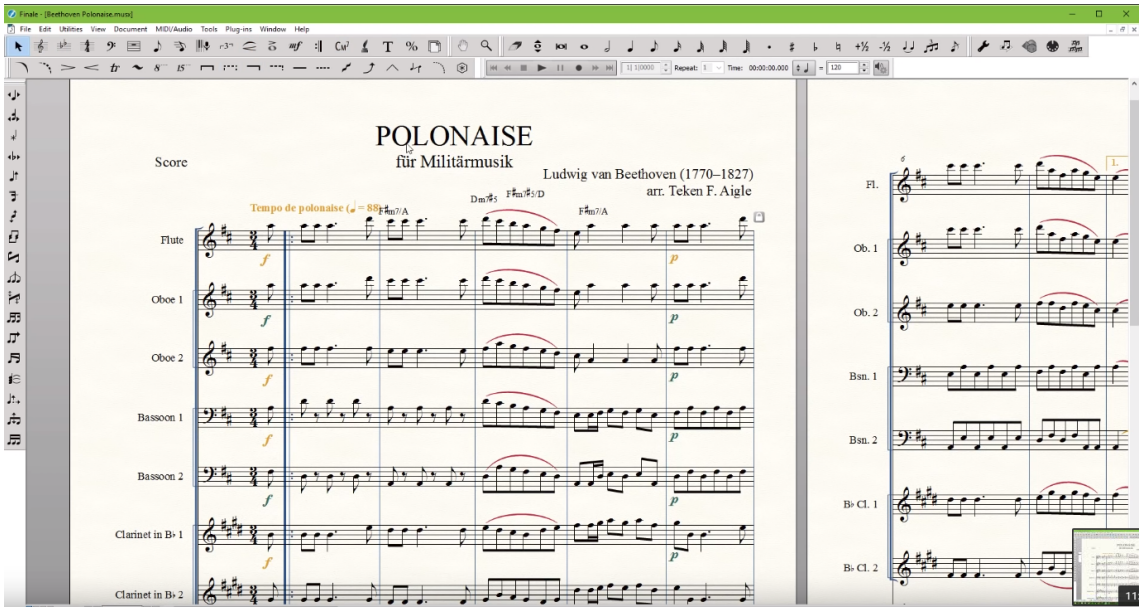


Figura 2.6: Captura de pantalla del programa Finale
Font: https://youtu.be/N9k___bNUh0s

2.3.3. Sibelius

Sibelius, [22] igual que Finale, és un programa amb llicència de pagament amb característiques molt semblants al seu principal competidor. Tot i que tots dos són programes d'edició de partitures digitals, Sibelius ha desenvolupat un millor aprenentatge ja que requereix menys temps per aprendre a escriure una partitura senzilla que Finale.

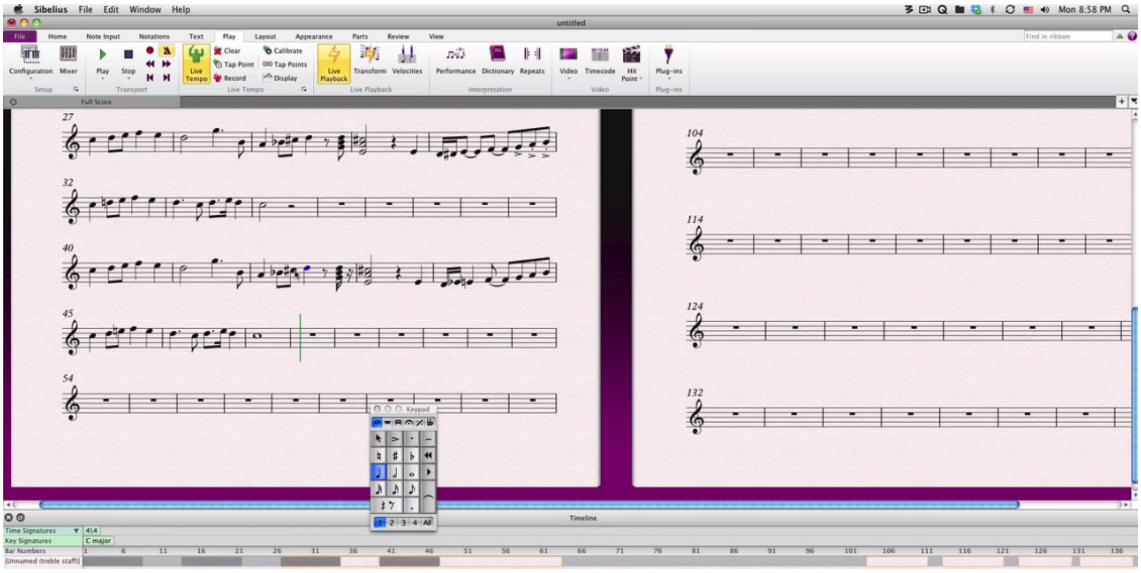


Figura 2.7: Captura de pantalla del programa Sibelius
Font: https://youtu.be/pBM-yXo0TM4

2.3.4. LilyPond

LilyPond [23] és un programa de software lliure d'edició de partitures per tots els sistemes operatius. A diferència dels altres tres programes, LilyPond no inclou interfície gràfica, sinó que funciona per línia de comandes. La documentació del programa és molt bona i necessària, ja que el seu aprenentatge és poc intuïtiu. Pot ser integrat en LaTeX entre altres sistemes d'edició de text.

En aquest exemple mostro l'escriptura del codi LilyPond i el seu resultat després de compilar:

```
1 \relative c'' {
2   \time 4/4
3   \clef treble
4   a1 b2 c4 d8 d8 d8 d8 e16 e16 e16 e16 e16
5 }
```

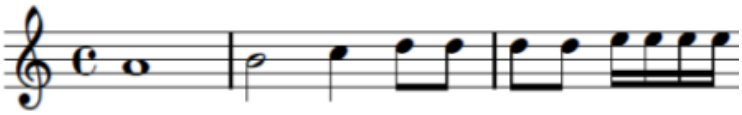


Figura 2.8: Representació gràfica del codi anterior **Font:** Lilypond.org

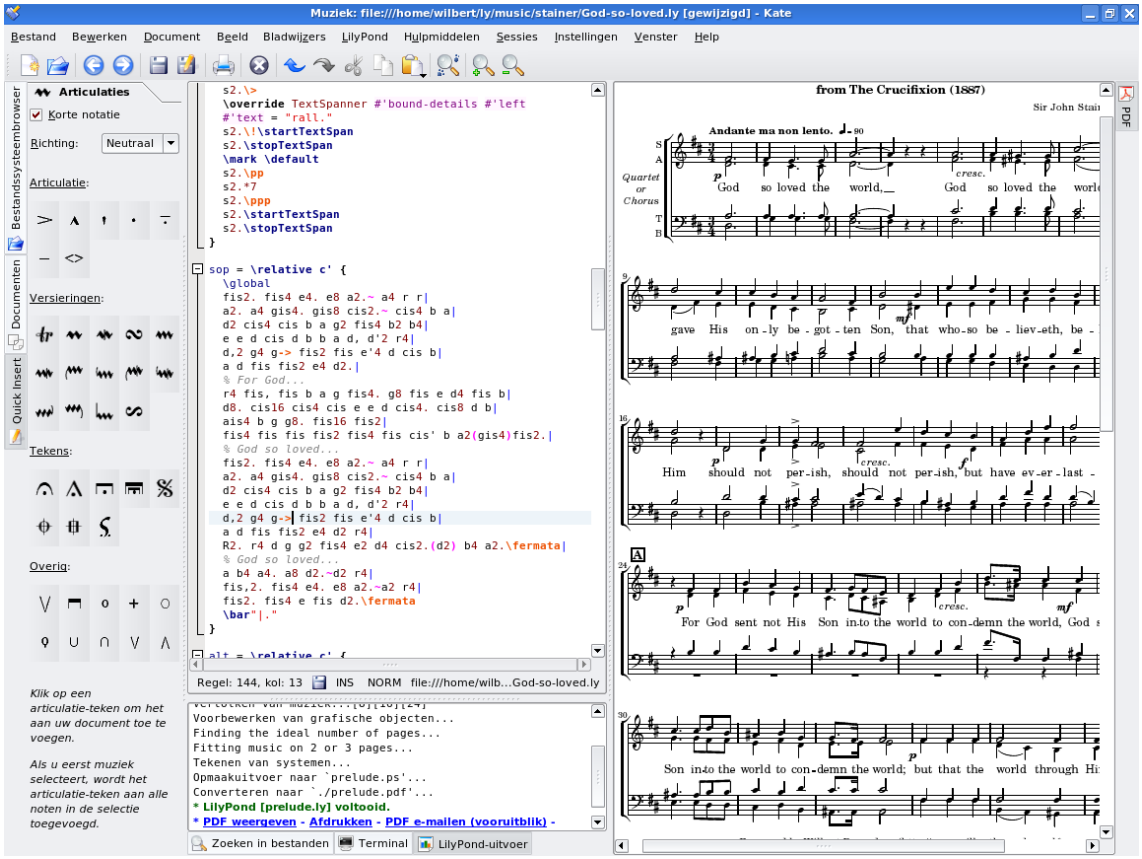


Figura 2.9: Captura de pantalla del programa Lilypond
Font: http://lilypond.org/pictures/lilykde-screenshot.png

2.4. Conclusions

Durant les últimes dècades s'ha augmentat la recerca i el desenvolupament de sistemes OMR. La detecció musical de caràcters és un camp de la informàtica que s'estén a moltes àrees com ara el reconeixement de patrons, la representació del coneixement, codificacions probabilístiques i detecció i correcció d'errors. En definitiva, l'OMR és un camp complex on apareixen diversos camps de la informàtica.

S'han presentat diversos programes OMR comercials, però cap d'ells amb un rendiment satisfactori, en particular, per a puntuacions de música manuscrita. Fins ara, fins i tot els productes de reconeixement més avançats ,(Notescan, Midiscan, Photoscore, Sharpeye), són incapaços d'identificar tots els símbols musicals. A més, aquests productes se centren principalment en el reconeixement de documents de música tipogràfica i impresa, i tot i que poden aconseguir resultats bastant bons per a aquests documents, no funcionen molt bé amb la música escrita a mà. L'estructura bidimensional de la notació musical revelada per la presència de les línies del pentagrama juntament amb l'existència de diversos símbols combinats organitzats al voltant dels caps de notes representa un alt nivell de complexitat en la tasca de l'OMR.

Un sistema OMR òptim ha de permetre, entre altres:

- Ser un mètode d'escriptura automàtica i d'estalvi de temps per convertir les partitures manuscrites a ordinador.
- Proporcionar l'accés a la música a persones sense coneixement musical previ.
- Aportar noves funcionalitats tecnològiques interactives com l'associació de partitures: detectar els plagis.
- Preservar el patrimoni cultural: per exemple per no perdre totes les partitures antigues escrites de forma manuscrita que amb el pas dels anys es desgasten i es tornarien impossibles d'entendre.

CAPÍTOL 3

Organització del projecte

3.1. Introducció

He volgut realitzar aquest projecte, entre altres motius, per permetre un accés a la música per tota la societat. Independentment de si tenen coneixement musical previ o no, totes les persones que utilitzin el meu programa podran escoltar acústicament el què hi ha escrit a les partitures. Recordar que l'objectiu general és aconseguir una aplicació que reconegui la música escrita en una partitura i la reprodueixi en format àudio.

El primer pas va ser buscar idees i aplicacions semblants per les quals tenir una idea de com començar a programar. Després d'un temps de cerca, vaig descobrir llibreries i tècniques de reconeixement basades amb el llenguatge de programació *Python* i va ser quan em vaig iniciar a escriure el codi.

He anat escrivint la memòria escrita a mesura que he anat avançant amb escriure el codi de l'aplicació de reconeixement. Un cop he aconseguit un programa capaç de reconèixer unes certes partitures amb característiques similars, he optat per crear un servidor web on bolcar-hi l'aplicació i fer-la accessible al públic general de forma lliure. Finalment he realitzat tests amb diferents partitures per analitzar subjectivament el tant per cent d'encert que he aconseguit implementar a l'aplicació de reconeixement de partitures, Music Scanner.

3.2. Cost del projecte

Per calcular el cost del projecte he tingut en compte dos conceptes: les despeses materials i les despeses personals.

3.2.1. Cost material

Per la realització del projecte he utilitzat un ordinador portàtil *HP Pavilion 15-N263ES* amb sistema operatiu *Ubuntu 16.04*. Suposant que la vida útil dels ordinadors portàtils és de 6 anys (6*12 = 72 mesos), tenint en compte que he estat 4'5 mesos amb el projecte i que el preu del portàtil és de 480€, equival a un preu de 30€.

També cal destacar que l'aplicació resultant la podrà utilitzar lliurement qui vulgui des d'una pàgina web, i que el preu dels sistemes OMR comercials oscil · len entre els 12 i els 200 euros.

3.2.2. Cost personal

Per calcular les despeses personals, he compatibilitzat el total d'hora invertides en la realització del treball. En total he necessitat 170 hores per la realització del treball. Tenint en compte que un enginyer informàtic cobra 50 euros/hora, el preu total és de 8500 €.

NOTA: Per tenir un pressupost complet també caldria incloure l'electricitat, la connexió a Internet i els desplaçaments a la universitat per les tutories de seguiment del treball.

3.3. Calendari: fases i durada

Vaig començar a realitzar el treball a principis del mes de febrer de 2018 i l'he finalitzat a principis de juny del mateix any. Han estat gairebé 5 mesos d'un intens treball en els que he creat l'aplicació , el servidor web i la memòria escrita. A continuació mostro un calendari general de les diferents fases i la durada d'aquestes.

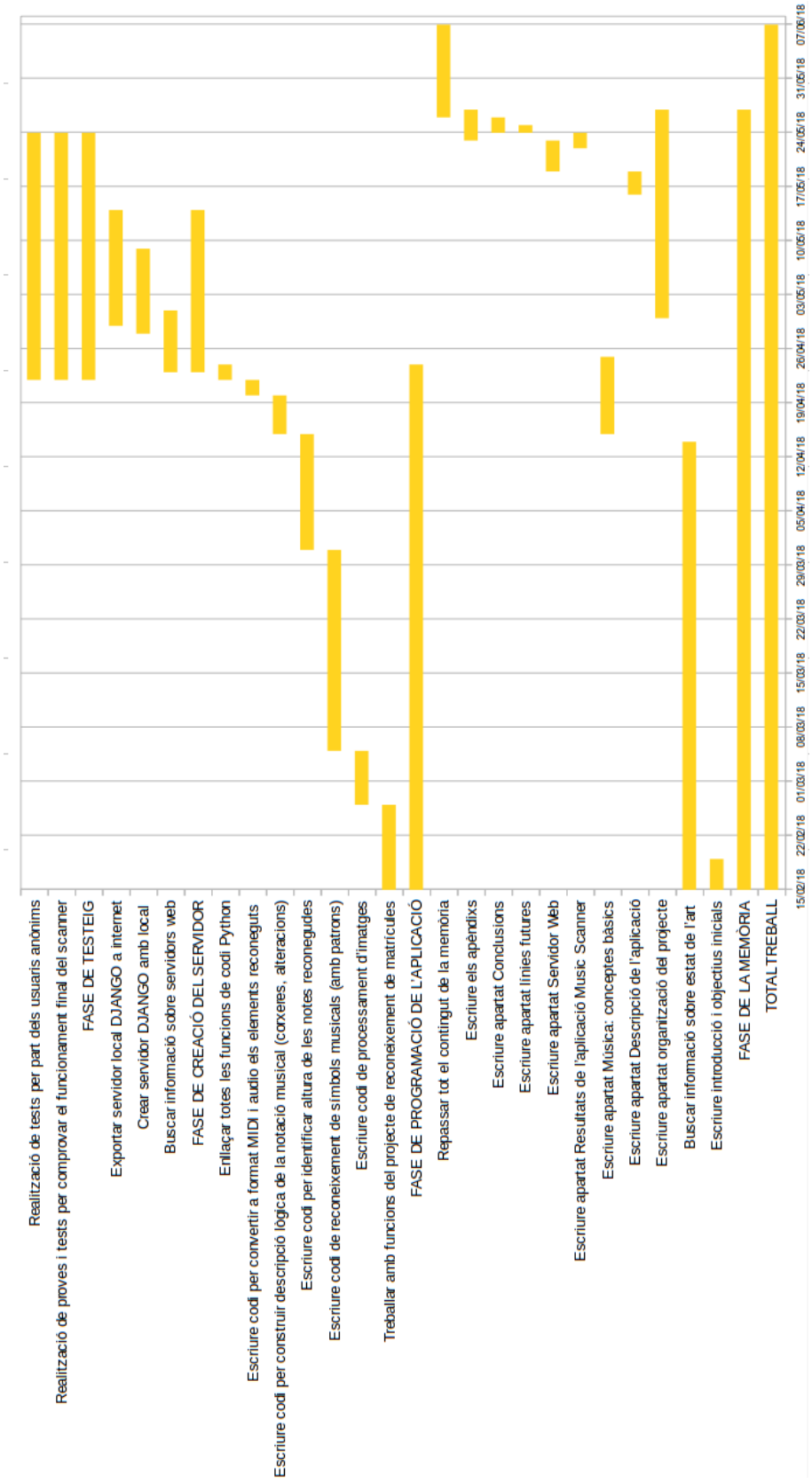


Figura 3.1: Calendari de fases del projecte **Font:** Pròpia

CAPÍTOL 4

Descripció de l'Aplicació

4.1. Introducció

Per escriure l'aplicació de reconeixement he partit d'un objectiu final concret i gràcies a la documentació i tota la informació que vaig estudiar en l'apartat referent a l'estat de l'art he anat avançant amb la programació.
A partir d'un objectiu general, he subdividit aquest objectiu amb petites parts, objectius específics, i aquests, en objectius més simples per tal d'anar evolucionant progressivament en la consecució dels objectius.

4.1.1. Idea inicial

Al principi volia aprofitar certes funcions del projecte de reconeixement de matrícules de cotxe de l'assignatura Informàtica de Q1. Em plantejo crear patrons de cada element musical, retallats i escalats de la mateixa mida pel reconeixement d'aquests. Trobo bases de dades amb patrons d'elements musicals que em guardo per utilitzar-los de patrons i començo a practicar amb les funcions del mòdul *Match* per reconèixer els patrons de les notes negres de l'escala natural de Do Major.

Després d'una reunió de seguiment del treball i de diversos intents d'avançar amb el reconeixement seguint el mètode del projecte de les matrícules sense aconseguir resultats positius, decideixo investigar més sobre com reconèixer elements musicals, (OMR), fins a trobar casos que utilitzen la llibreria OpenCV [3]. Així doncs, co-

menço a estudiar a fons i provar exemples senzills amb funcions de OpenCV per aprendre'n el funcionament.

4.2. Definició de l'estàndard

Actualment existeixen una infinitat de partitures en una multitud de formats. És per això molt important, establir unes bases o requisits previs per filtrar aquelles partitures per les quals l'aplicació no és capaç de reconèixer i analitzar els símbols musicals escrits.
El fet de ser un treball de final de grau, cal destacar que l'aplicació no té una capacitat de reconeixement comparable als sistemes comercials.

La llista següent defineix una partitura estàndard amb la qual es pot assegurar un alt tant per cent d'encert de l'aplicació:

- Imatge d'una partitura escrita a format digital, no manuscrita
- Imatge amb mides superior als 1000px i inferior als 2000px d'amplada
- Partitura d'una sola línia melòdica
- Reconeixement dels ritmes: rodona, blanca, negra i corxeres (sempre i quan les corxeres estiguin ajuntades i no una corxera sola)
- Reconeixement de notes entre el si3 i do6
- No reconeixement de cap tipus de silenci
- Reconeixement de totes les alteracions de l'armadura
- Reconeixement de partitures amb clau de sol
- Reconeixement de partitures amb compassos simples

- La imatge de la partitura ha d'estar amb les línies de pentagrama perfectament horitzontals
- Ha d'existir un marge en blanc, de la mida aproximada de les cinc línies de pentagrama entre els extrems inferior i superior de la imatge

Tots els elements que no estiguin inclosos a la llista de l'estàndard no es tindran en compte com a elements no reconeguts a l'hora de realitzar els tests de l'aplicació.

4.3. Llibreries utilitzades

Per escriure l'aplicació he utilitzat les següents llibreries pel llenguatge de programació *Python*, versió 3 o superior:

- OpenCV
- NumPy
- Sys
- MIDIUtil
- PySynth

4.3.1. OpenCV

OpenCV (Open Source Computer Vision Library) [3], actualment a la versió CV2, és una llibreria amb llicència BSD [24], que significa que qualsevol pot utilitzar i modificar el codi lliurement. La llibreria compta amb més de 2500 algorismes optimitzats, que inclouen un conjunt complet d'algorismes d'aprenentatge automàtic de la visió per computadora. Aquests algorismes es poden utilitzar per detectar i reconèixer cares, identificar objectes, classificar accions humanes en els vídeos, etc. A més de Python, la llibreria OpenCV està present a altres llenguatges com C++, Java, MATLAB i suporta les interfícies Windows, Linux, Android i Mac OS.

Els algorismes que he utilitzat d'aquesta llibreria són els de detecció i reconeixement de patrons en imatges (explicat al punt 4.6) i els de transformació de les mides i colors de les imatges (explicat al punt 4.5) .

4.3.2. NumPy

NumPy [4] és una paquet de Python que proporciona suport de càlcul per vectors i matrius, constituint així una llibreria de funcions matemàtiques d'alt nivell per operar amb aquests vectors o matrius. En l'aplicació utilitzo la funció *where()* per retornar les posicions dels valors de result els quals superen un valor llindar:

```
result = np.where(result >= llindar)
```

4.3.3. Sys

El mòdul Sys [25] proporciona accés a algunes variables utilitzades per l'intèrpret i per a funcions que interactuen fortament amb l'intèrpret. L'intèrpret, terminal, o línia de comandes, és un programa informàtic capaç d'analitzar i executar altres programes.

4.3.4. MIDIUtil

MIDIUtil [5], és una llibreria de llenguatge *Python* que permet escriure fitxers Multimèdia d'Interfície Digital d'Instruments Musicals, coneguts amb el nom abreviat de MIDI. És una llibreria orientada a objectes. En el meu programa utilitzo aquesta llibreria per transformar els elements musicals reconeguts de la partitura a notes en format MIDI.
Aquest programari es distribueix sota una llicència de codi obert.

4.3.5. PySynth

El mòdul PySynth [26] és un conjunt de sintetitzadors de música de codi obert i escrits en Python 3. L'objectiu és convertir els arxius escrits en format ABC o MIDI

a fitxers WAV. L'utilitzo a la última línia del codi, quan vull convertir el fitxer MIDI a àudio.

4.4. Arquitectura de l'aplicació

L'aplicació consta de 4 mòduls *Python* relacionats entre ells, i dels quals se'n poden diferenciar 5 tasques:

1. **Processament de la imatge:** adaptar la imatge d'entrada (amb format jpg, jpeg o png) per garantir el funcionament de les següents tasques.

2. **Reconeixement dels patrons:** Comparar els patrons existents a la base de dades amb la imatge de la partitura processada i obtenir-ne les coincidències d'aquests patrons dels elements musicals dins la partitura.

3. **Reconeixement del to de les notes:** Un cop reconeguts tots els elements musicals de l'estàndard, passem a reconèixer l'alçada de cada nota amb l'objectiu de diferenciar si una negra, blanca o rodona és un do,re,mi,fa,sol,la o si i de quina octava es tracta.

4. **Imposar les regles musicals:** Analitzar la partitura perquè compleixi les regles bàsiques de les alteracions en la música.

5. **Conversió de format:** Transformar els objectes reconeguts a un format adequat per la seva representació de forma acústica i gràfica.

A la figura 4.1 mostro la relació entre les diferents tasques que realitza l'aplicació.

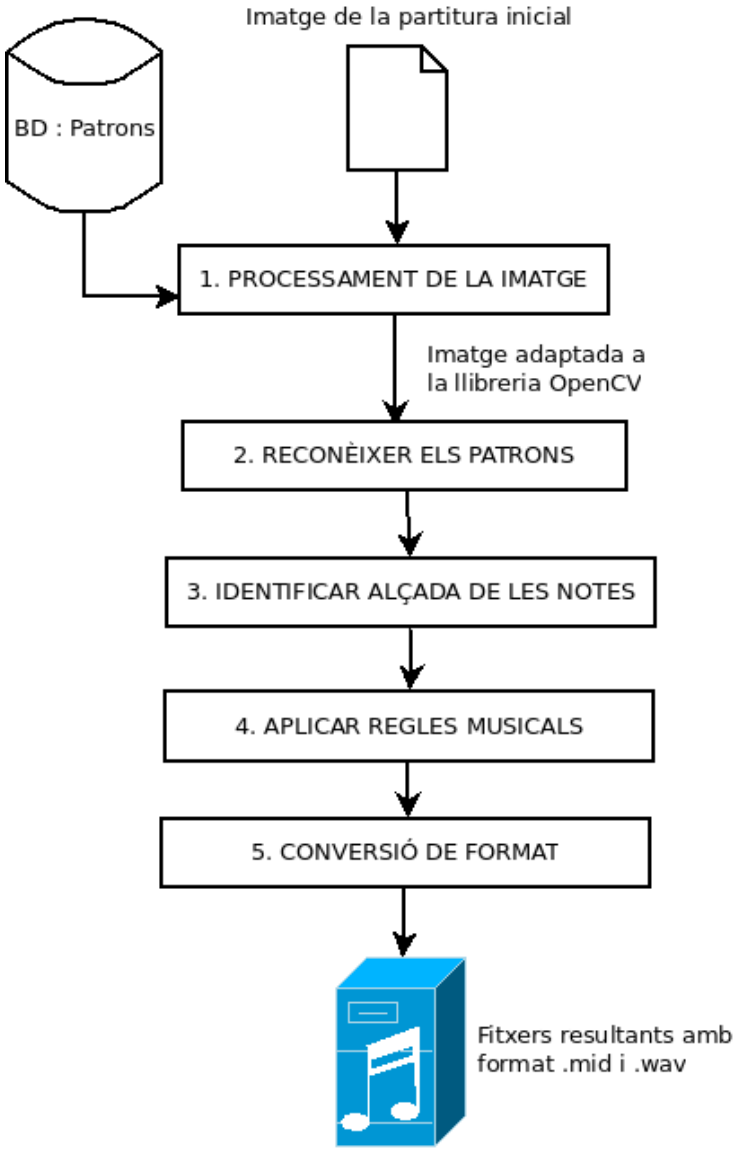


Figura 4.1: Arquitectura de l'aplicació **Font:** Elaboració pròpia.

A la següent figura mostro la relació entre els diferents mòduls amb què s'estructura l'aplicació.

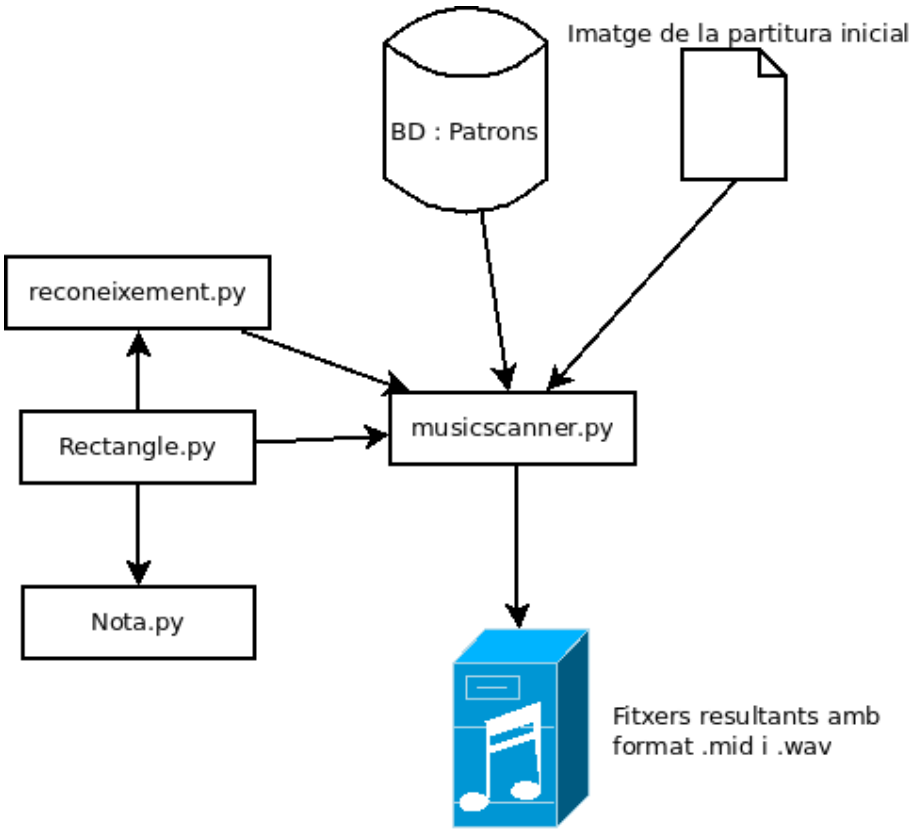


Figura 4.2: Esquema de mòduls **Font:** Elaboració pròpia.

4.5. Processament de la imatge

Aquesta és la primera etapa de l'aplicació. Per garantir el funcionament de les funcions de les següents tasques, cal adaptar la imatge que es vol escanejar a un format llegible per l'aplicació. He optat per la llibreria OpenCV [3] per dur a terme les tasques del maneig de les imatges, per tant, cal llegir la imatge amb la funció *imread()* del mòdul OpenCV.

Amb els patrons també els passo per un processat per què siguin llegibles quan els comparo amb la imatge de la partitura.

La figura 4.3 mostra els dos processaments que es duen a terme per integrar la imatge i els patrons dins l'aplicació:

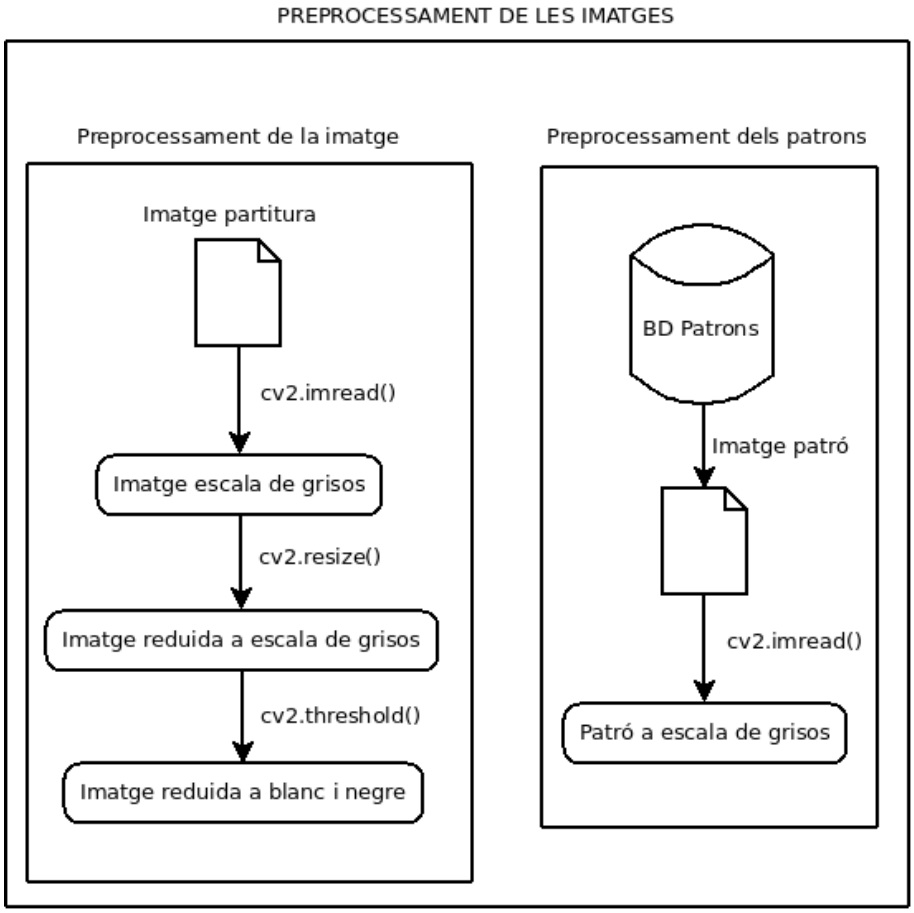


Figura 4.3: Diagrama de blocs de la primera etapa **Font:** Elaboració pròpia.

La imatge de la partitura la transformo a escala de grisos. És el pas previ necessari per la transformació a blanc i negre (binarització). També li redueixo la mida en cas que la seva amplada sigui superior a 1600px ja que per una mida superior, l'aplicació tarda més estona en executar-se completament i els patrons necessitarien augmentar la mida i perdrien resolució.

4.5.1. Conversió a escala de grisos

Per convertir les imatges a escala de grisos he utilitzat la funció *cv2.imread()*, que a part de convertir-la a escala de grisos, la integra al format de la llibreria OpenCV. Com a primer argument se li ha de passar el nom de la imatge o la ruta de la imatge completa, en cas que no estigui dins el mateix directori de treball. El segon argument és un indicador que especifica la manera de llegir la imatge. Pot valdre 1, 0 o -1 en

funció de:

- cv2.IMREAD_COLOR: carrega una imatge en color. Qualsevol transparència d'imatge no es tindrà en compte. És la opció predeterminada.
- cv2.IMREAD_GRAYSCALE: carrega la imatge en escala de grisos
- cv2.IMREAD_UNCHANGED: carrega la imatge com a tal incloent el canal alfa

En el meu cas li he escrit un 0 perquè vull que la llegeixi en escala de grisos:

```
img = cv2.imread(img_partitura, 0)
```

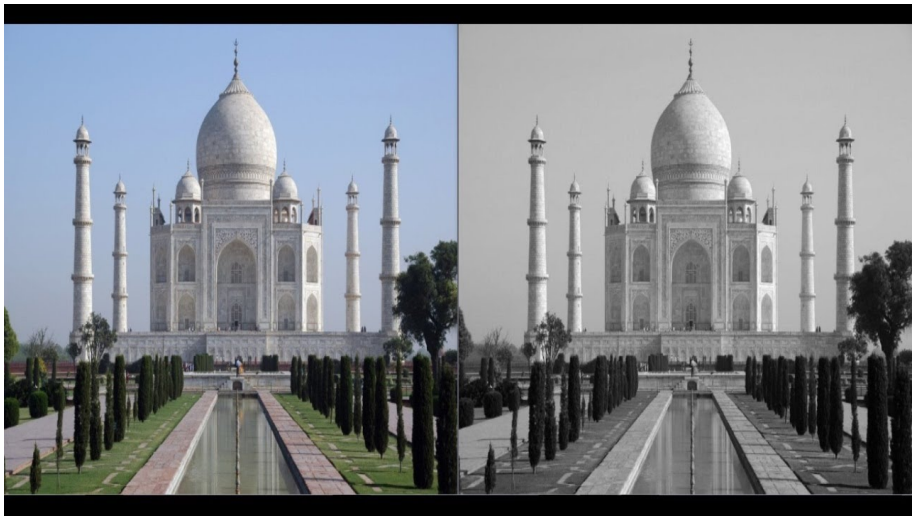


Figura 4.4: Conversió de RGB a escala de grisos
Font: https://i.ytimg.com/vi/zSo3QZTleqA/maxresdefault.jpg.

Cal tenir en compte que si la ruta de la imatge està equivocada, la funció no avisarà d'un error, sinó que es guardarà un valor indefinit com a imatge.

4.5.2. Canviar la mida d'una imatge

Utilitzo la funció *cv2.resize()* per canviar la mida de les imatges i dels patrons segons s'escaigui. La mida de la imatge *src* pot augmentar o disminuir segons la proporció dels eixos *(fx,fy)* que li escric a l'argument de la funció.

```
img_escalada = cv2.resize(src, None,
        fx = escalat, fy = escalat, interpolation = cv2.INTER_AREA)
```

Existeixen cinc tipus diferents d'interpolació. He escollit el mètode *cv2.INTER_AREA*, perquè segons l'estudi comparatiu [27], quan aquest mètode s'utilitza per disminuir la mida de les imatges, és el que aconsegueix unes vores més suaus, per una posterior binarització.
He cregut que és el millor mètode a utilitzar ja que l'utilitzo bàsicament per disminuir les imatges i, tant per velocitat d'execució com per qualitat dels resultats, he cregut que era el més adequat.

4.5.3. Conversió a blanc i negre

La conversió a blanc i negre, també anomenat *binarització*, és un mètode de transformació dels píxels d'una imatge en escala de grisos a píxels blancs i negres. Binarització prové del fet que només hi ha dos tipus de píxels possibles (blanc o negre). La funció *cv2.treshold()* de la llibreria OpenCV [3] et modifica una imatge que estigui amb escala de grisos en una imatge binaritzada.

Tenint en compte que no es pot establir un valor estàndard per totes les imatges a binaritzar, utilitzo el mètode d'Otsu [17] per calcular el llindar òptim de cada imatge. Aquest mètode, com ja he explicat a 2.2.1, calcula el valor òptim amb el qual discernir entre si un píxel és blanc o és negre.

```
ret,img_bw = cv2.threshold(img_escalada,127,255,
                cv2.THRESH_BINARY+cv2.THRESH_OTSU)
```

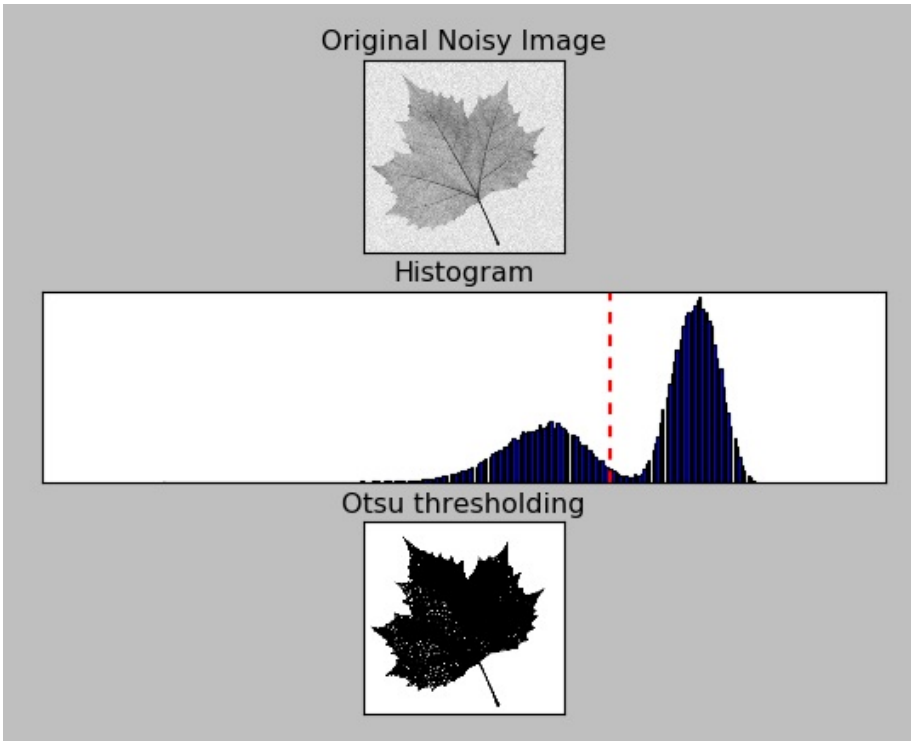


Figura 4.5: Binarització d'una imatge aplicant el llindar d'Otsu
Font: https://www.meccanismocomplesso.org/en/

Un cop arribat en aquest punt, la imatge de la partitura ja està preparada per aplicar-hi el reconeixement de patrons.

4.6. Reconeixement dels patrons

L'etapa de reconeixement dels patrons segueix el mateix esquema independentment del patró de l'element musical. Només varia amb el reconeixement de les línies de la partitura, ja que és una ampliació de la part del reconeixement del pentagrama de la partitura.

Com mostro a la figura 4.6, pel reconeixement de patrons he creat tres funcions amb tasques diferenciades que explico a continuació.

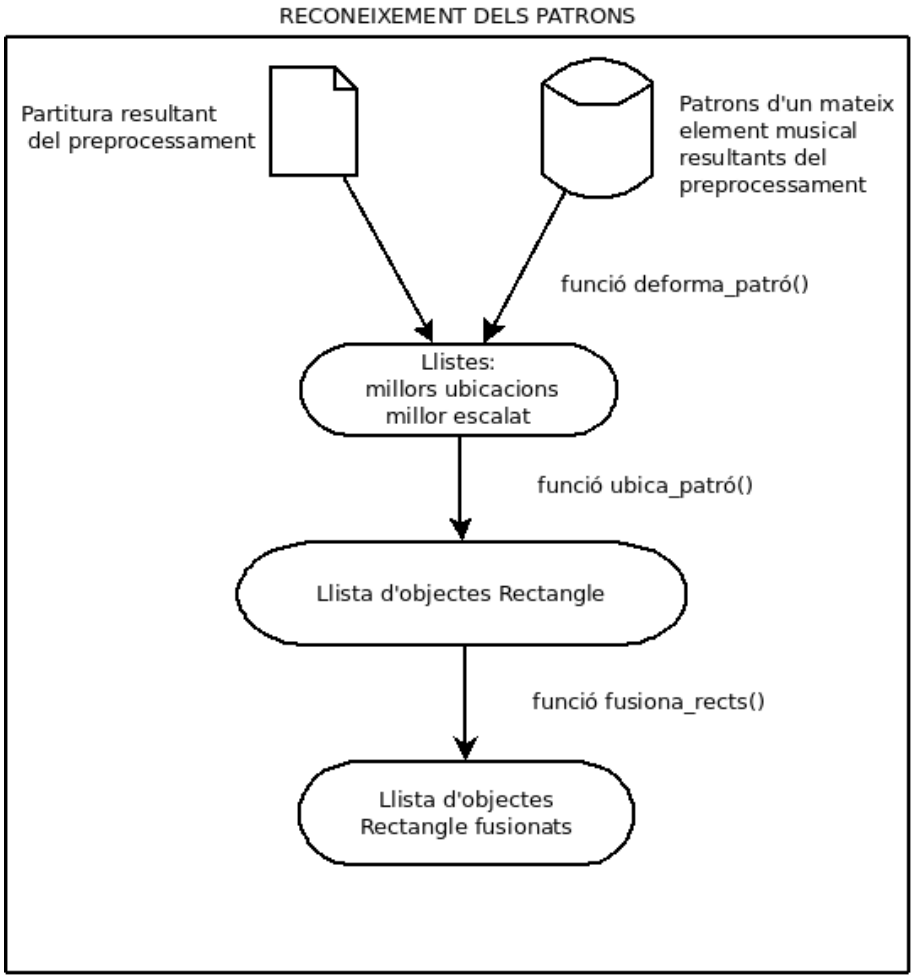


Figura 4.6: Etapa de reconeixement dels patrons **Font:** Pròpia

4.6.1. Deformar els patrons

He creat una funció per deformar els patrons, deforma_patro() que se li passa com a paràmetre els següents arguments:

- **img:** La imatge de la partitura resultant del preprocessament.
- **patrons:** Els patrons d'un mateix element musical resultants del preprocessament.
- **start_percent:** Valor escalat mínim.
- **stop_percent:** Valor escalat màxim.

- **llindar:** valor llindar

Els elements musicals poden variar en mida i tipus de lletra per cada partitura en concret. Tenint en compte aquest fet, aquesta funció varia la mida de diferents patrons ,(amb la funció cv2.resize() 4.5.2), des del valor *start* fins al valor *stop* i per cada mida del patró, crida la funció *cv2.matchTemplate()* per determinar si una plantilla està present o no a la imatge.
Cada coincidència d'un patró amb la imatge, es compara amb el valor *llindar*. Si la coincidència és inferior al valor *llindar* la descartem. En cas contrari, la guardem.
D'aquesta manera es coneix el millor patró a partir de de les coincidències entre patró/imatge i les millors ubicacions dels patrons en la partitura.
Els valors start/stop/llindar s'han ajustat a partir d'experimentar amb diferents imatges i definir una mida màxima de 1600px d'amplada i comprovar que són els valors que encaixen més.

La funció retorna un llistat de llistes corresponents a les ubicacions i els escalats que causen els millors resultats (millors coincidències) amb la imatge.

Funció cv2.matchTemplate()

La llibreria *OpenCV* disposa de la funció cv2.matchTemplate() [28] utilitzada per detectar la concordança de patrons en una imatge més gran. El que fa aquesta funció és fer lliscar el patró per tota la imatge amb l'objectiu de trobar-hi coincidències.

4.6.2. Ubicació del patró a la partitura

La funció *ubica_patro()* és el pas següent de la detecció dels patrons. Aquesta, a partir dels resultats de la funció *deforma_patró()*, genera els objectes de la classe Rectangle, 4.10.1, corresponents als patrons reconeguts a la partitura i en retorna la llista de rectangles.

4.6.3. Fusionar patrons

La funció fusiona_rects() és la última de l'etapa del reconeixement de patrons. Utilitza funcions de la classe Rectangle per fusionar els objectes Rectangle que se sobreposen a partir d'un valor *llindar*. Aquest valor, tot i anomenar-se *llindar* igual que a les altres dues funcions, és de valor diferent ja que efectua una altra funcionalitat.

La funció fusiona_rects() retorna la llista definitiva d'objectes Rectangle per avançar fins la següent etapa: Reconeixement del to de les notes.

4.6.4. Identificar línies de la partitura

Per reconèixer les línies de la partitura, com he dit anteriorment, es realitzen unes tasques extres amb la finalitat d'obtenir els objectes Rectangle corresponents a cada línia de la partitura.

A partir de l'alçada del rectangle corresponent al patró reconegut del pentagrama, obligo a totes les línies de la partitura a tenir aquest mateix valor d'alçada. Això em permetrà tenir el mateix algorisme de reconeixement del to de les notes, que explico a la següent secció.

4.6.5. Exemple de reconeixement de patrons

A continuació, a partir de la partitura "Dins la Fosca"mostro els resultats obtinguts amb el reconeixement dels patrons d'elements musicals:

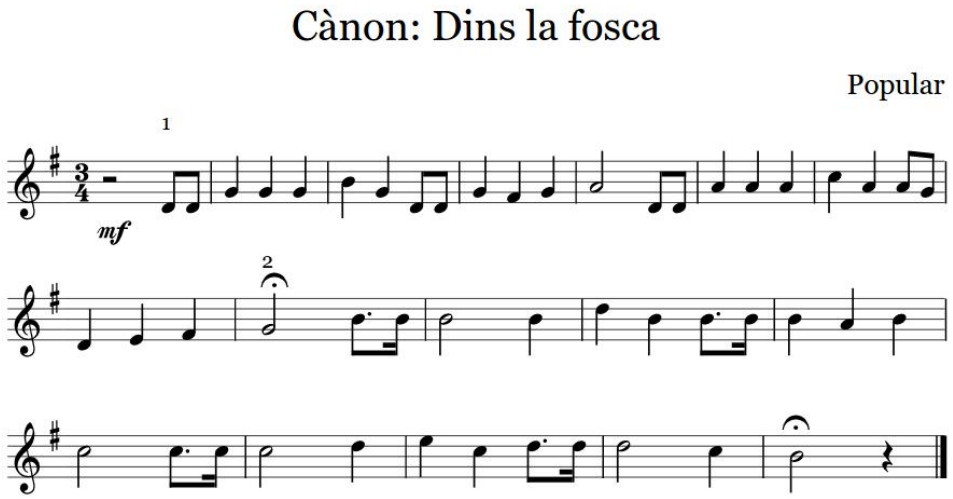


Figura 4.7: Partitura Dins la Fosca
Font: sites.google.com/site/wikimusiquem/recursos-tic/flauta/partitures

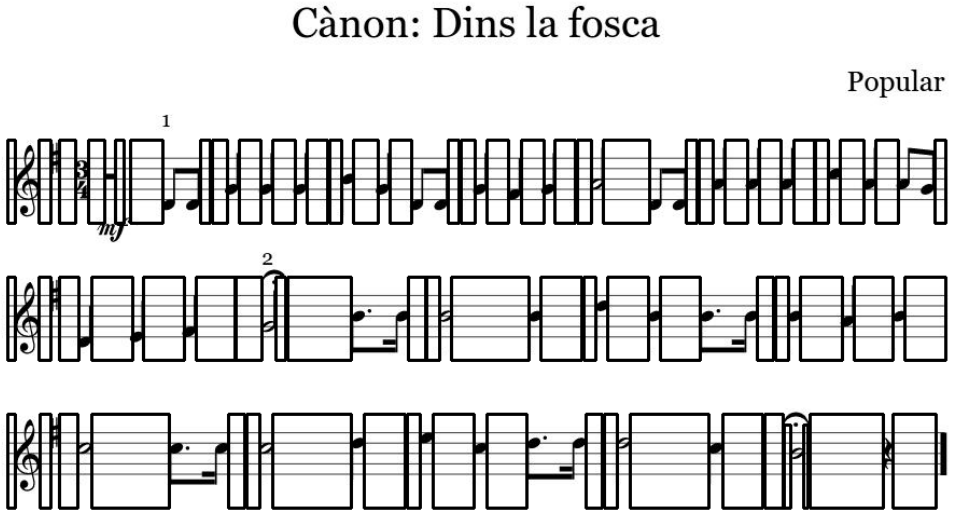


Figura 4.8: Resultat reconeixement patrons pentagrama **Font:** Pròpia

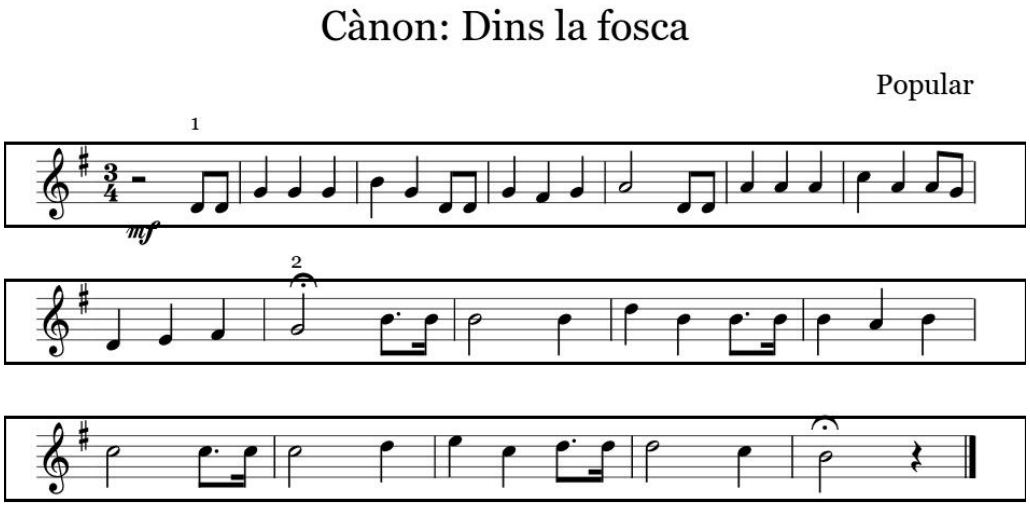


Figura 4.9: Resultat reconeixement línies pentagrama **Font:** Pròpia

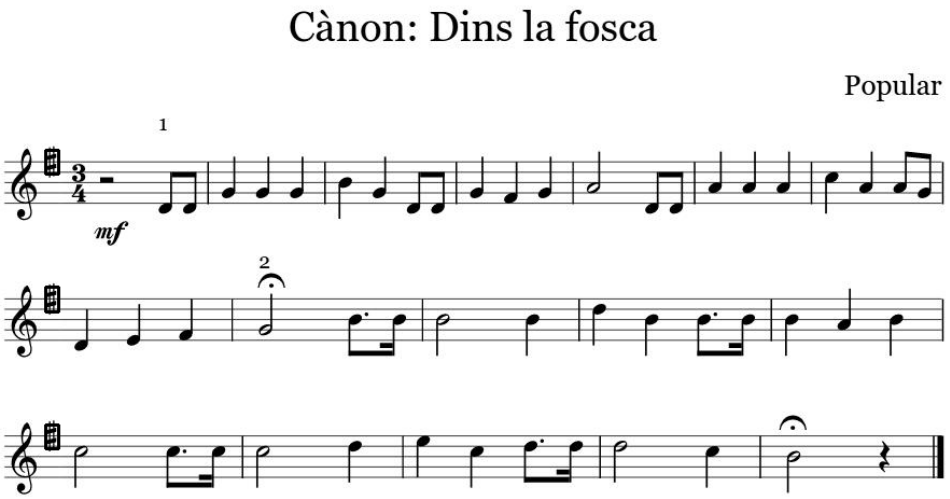


Figura 4.10: Resultat reconeixement patrons sostinguts **Font:** Pròpia

Com que la partitura no té cap bemoll, la figura 4.11 és la mateixa que la figura de la partitura inicial 4.7.

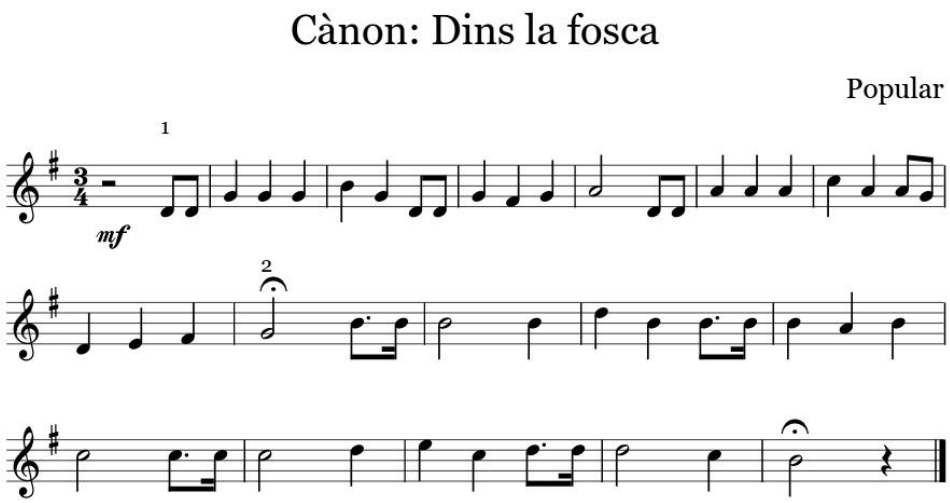


Figura 4.11: Resultat reconeixement patrons bemolls **Font:** Pròpia

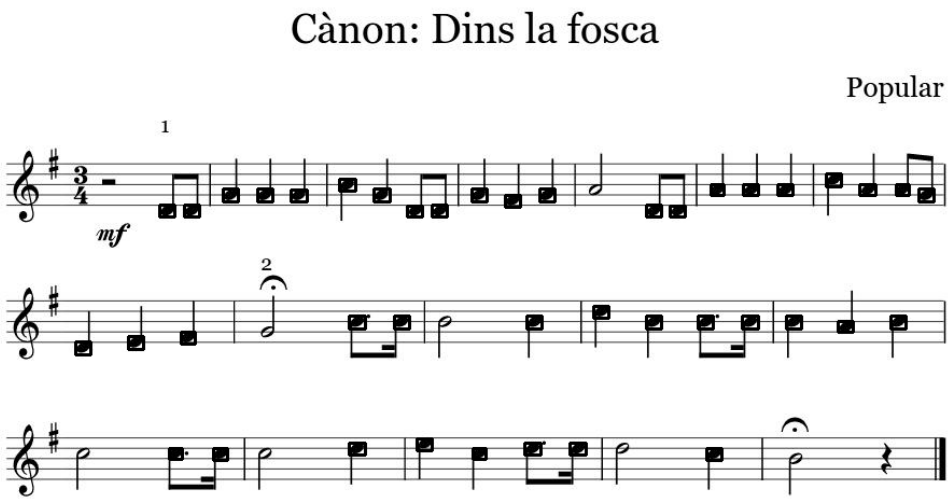


Figura 4.12: Resultat reconeixement patrons negres **Font:** Pròpia

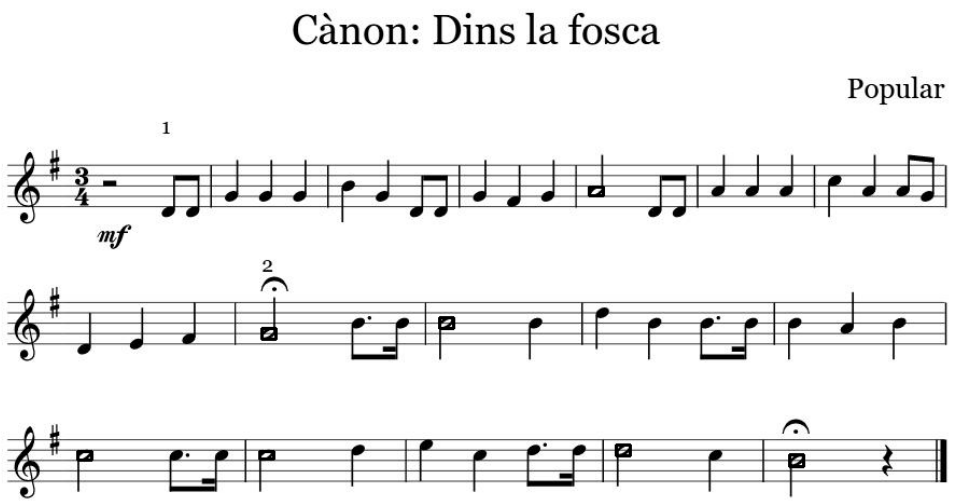


Figura 4.13: Resultat reconeixement patrons blanques **Font:** Pròpia

A la lletra *"s"* del títol Dins la Fosca, l'aplicació ha detectat una rodona. Serà feina de l'etapa de reconeixement del to de les notes quan decidirà descartar aquesta coincidència perquè es troba fora dels límits dels rectangles de les línies de la partitura.

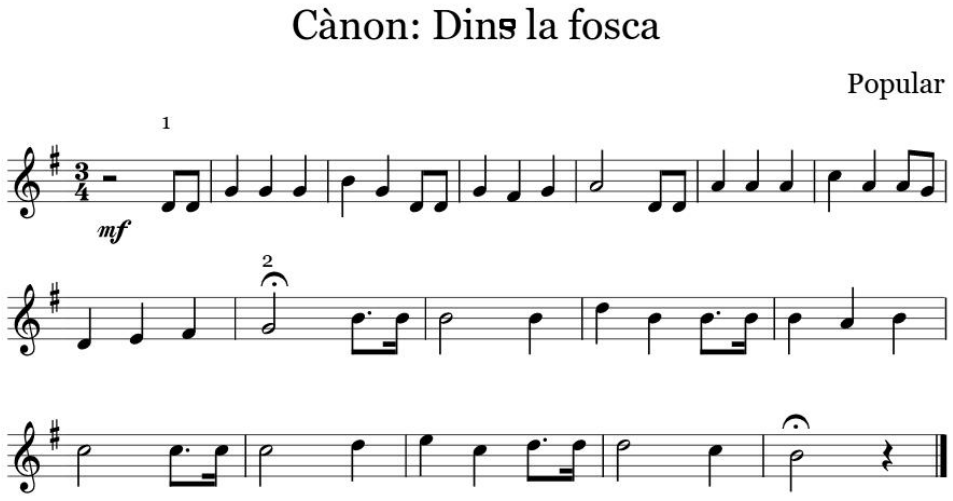


Figura 4.14: Resultat reconeixement patrons rodones **Font:** Pròpia

4.7. Reconeixement del to de les notes

Per reconèixer el to de les notes utilitzo la classe Nota. El procés és el mateix per a cada línia de la partitura: Miro tots els objectes Rectangle de cada element musical. Per tots els elements musicals Rectangle que estan dins de l'objecte Rectangle de la mateixa línia de la partitura, es crea un objecte Nota corresponent a aquest element musical en concret. La resta d'elements musicals s'afegiran quan el codi miri les altres línies de la partitura o, en cas que hi hagi un element musical fora de les línies de pentagrama, l'ignorarà com en el cas de la rodona amb la partitura de Dins la Fosca, figura: 4.14.

La classe Nota utilitza un algorisme de reconeixement del to de les notes basat en la posició del rectangle d'aquestes dins del rectangle de la línia de la partitura. D'aquesta manera, segons la posició y del rectangle de l'element musical, calculo el to de la nota.

Per exemple, amb la partitura *Oh, Susana !*, mostraré com es reconeixen els tons de les primeres notes:

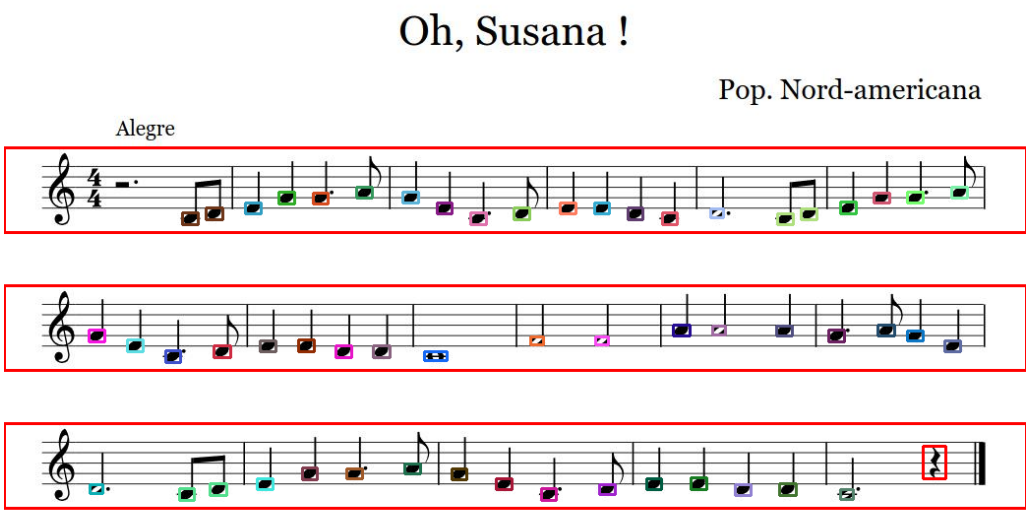


Figura 4.15: Partitura *Oh, Susana !* amb els objectes Rectangle ja dibuixats
Font: Pròpia

L'algorisme que identifica el to de les notes, a partir de la posició central del rectangle de cada nota, mesura la distància entre aquest centre i la línia superior del rectangle de la línia de la partitura (línies negres).Per veure el codi de l'algorisme el podeu trobar a l'apèndix E.4. A continuació i, sabent que per una alçada h (línia verda) de 87,6px poden aparèixer un total de 17 notes diferents, i com a màxim hi poden haver 20 notes, calcula el nombre total de notes que poden existir a la partitura.

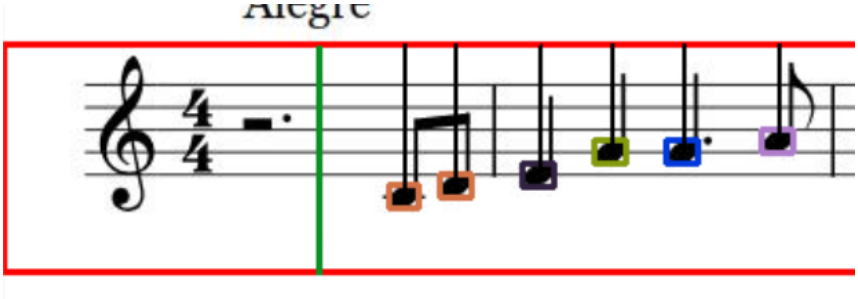


Figura 4.16: Inici de la partitura anterior **Font:** Pròpia

En aquest cas, l'alçada *h* és de 86.14px , per tant poden existir 17 notes en totes les línies o espais del pentagrama. El següent pas és calcular el valor *step* que servirà

per localitzar la posició corresponent al to de les notes.

$$step = \frac{h}{total\_notes} \tag{4.1}$$

Amb l'exemple de *Oh, Susana !*, el valor step és

$$step = \frac{h}{total\_notes} = \frac{86,14}{17,5} = 4,92 \tag{4.2}$$

El nombre total_notes l'incremento 0,5 perquè amb moltes proves de partitures que he realitzat, algunes notes les reconeixia més greus del que eren, i incrementant en 0,5 el total de notes, aquest problema s'ha solucionat.

He creat un diccionari amb *Python*, on la clau és un nombre enter ordenat del 0 al 24 i el valor és una tupla amb el nom de la nota amb notació anglesa i el nombre resultant de la fórmula 4.3 corresponent al to de la nota en notació MIDI [29].

```
#Diccionari
nota_defs = {
      0 : ("c6", 84),
      1 : ("b5", 83),
      2 : ("a5", 81),
      3 : ("g5", 79),
      4 : ("f5", 77),
      5 : ("e5", 76),
      6 : ("d5", 74),
      7 : ("c5", 72),
      8 : ("b4", 71),
      9 : ("a4", 69),
      10: ("g4", 67),
      11: ("f4", 65),
```

```
    12: ("e4", 64),
    13: ("d4", 62),
    14: ("c4", 60),
    15: ("b3", 59),
    16: ("a3", 57),
    17: ("g3", 55),
    18: ("f3", 53),
    19: ("e3", 52),
    20: ("d3", 50),
    21: ("c3", 48),
    22: ("b2", 47),
    23: ("a2", 45),
    24: ("f2", 41)}
```

Conversió de freqüència a valor MIDI, on *"m"* és el valor MIDI corresponent a la freqüència de cada nota.

$$m = 69 + 12\log_2(\frac{f}{440Hz}) \tag{4.3}$$

El valor resultant del logaritme és el nombre d'octaves per sobre o per sota de la nota A4 (que té una freqüència de 440Hz). Al multiplicar-ho per 12 és per obtenir els semitons i la suma de 69 és el nombre m corresponent a la nota A4.

El següent pas per reconèixer les notes és aplicar amb cadascuna d'elles aquest *bucle for* per tal de conèixer-ne el to. Aquest fragment de codi busca que la posició de la nota estigui entre un valor step que el vaig incrementant amb la sentinella "*i*". Aquesta sentinella correspon a la clau del diccionari esmentat abans.

```
i = 1
for i in range(len(nota_defs)):
```

```
if (nota_real >= (i*step - step) and nota_real <= (i*step + step)):
  nota_def = nota_defs[i]
  break
else:
  i+=1
```

A la taula 4.1 mostro els resultats després de passar pel codi anterior, de les sis primeres notes de la partitura 4.16:

Taula 4.1: Taula de relació posició de la nota amb el to

centre_nota	72,7	67,7	61,7	50,7	51,2	45,7
nota	C4	D4	E4	G4	G4	A4
m [MIDI]	60	62	64	67	67	69
sentinella [i]	14	13	12	10	10	9

A la taula 4.2 mostro la relació entre freqüències i el seu valor corresponent MIDI:

MIDI number	Note name	Keyboard	Frequency Hz
21	A0		27.500
22			29.135
23	B0		30.868
24	C1		32.703
25			34.648
26	D1		36.708
27			38.891
28	E1		41.203
29	F1		43.654
30			46.249
31	G1		48.999
32			51.913
33	A1		55.000
34			58.270
35	B1		61.735
36	C2		65.406
37			69.296
38	D2		73.416
39			77.782
40	E2		82.407
41	F2		87.307
42			92.499
43	G2		97.999
44			103.83
45	A2		110.00
46			116.54
47	B2		123.47
48	C3		130.81
49			138.59
50	D3		146.83
51			155.56
52	E3		164.81
53	F3		174.61
54			185.00
55	G3		196.00
56			207.65
57	A3		220.00
58			233.08
59	B3		246.94
60	**C4**		**261.63**
61			277.18
62	D4		293.67
63			311.13
64	E4		329.63
65	F4		349.23
66			369.99
67	G4		392.00
68			415.30
69	**A4**		**440.00**
70			466.16
71	B4		493.88
72	C5		523.25
73			554.37
74	D5		587.33
75			622.25
76	E5		659.26
77	F5		698.46
78			739.99
79	G5		783.99
80			830.61
81	A5		880.00
82			932.33
83	B5		987.77
84	C6		1046.5
85			1108.7
86	D6		1174.7
87			1244.5
88	E6		1318.5
89	F6		1396.9
90			1480.0
91	G6		1568.0
92			1661.2
93	A6		1760.0
94			1864.7
95	B6		1975.5
96	C7		2093.0
97			2217.5
98	D7		2349.3
99			2489.0
100	E7		2637.0
101	F7		2793.0
102			2960.0
103	G7		3136.0
104			3322.4
105	A7		3520.0
106			3729.3
107	B7		3951.1
108	C8	J. Wolfe, UNSW	4186.0

Taula 4.2: Relació de les notes
Font: https://newt.phys.unsw.edu.au/jw/notes.html

4.8. Imposar les regles musicals

4.8.1. Alteracions

Un cop coneixem les alçades de les notes, el següent pas és comprovar si l'aplicació ha detectat alteracions. En cas afirmatiu, quantificar quantes alteracions ha detectat i si són sostinguts o bemolls, i a partir de l'ordre dels sostinguts i l'ordre dels bemolls, modificar el to de totes les notes que coincideixin amb les alteracions detectades.
Per exemple, si el sistema detecta 3 sostinguts, totes les notes F,C,G de qualsevol escala se'ls incrementarà la tonalitat un semitò.
Si el sistema detecta 3 bemolls, totes les notes B,E,A de qualsevol escala se'ls disminuirà la tonalitat un semitò.

4.8.2. Corxeres

Per diferenciar una negra d'una corxera, miro si entre dos objectes Rectangle corresponents al pentagrama, hi ha més d'un objecte Rectangle d'una negra. En cas afirmatiu, tots els objectes Rectangle d'una negra existents entre dos objectes Rectangle de pentagrama els converteixo a corxeres.
A la figura 4.17 mostro un fragment d'una partitura amb els objectes Rectangle de pentagrama dibuixats.

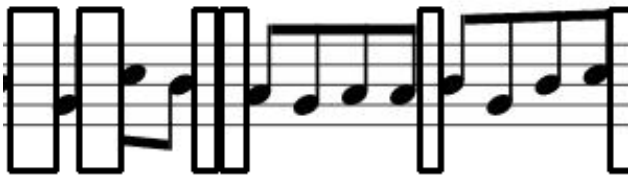


Figura 4.17: Reconeixement de corxeres **Font:** Pròpia

Com he explicat a l'estàndard de l'aplicació, l'aplicació només reconeix les corxeres ajuntades. Totes les corxeres escrites sense una barra que les ajunti entre elles no

les reconeixerà. De la mateixa forma que un grupet de semicorxeres el reconeixerà com a corxeres. És una limitació de l'aplicació escrita a l'estàndard 4.2.

Ordenats de 1 a 10 i d'esquerra a dreta tots aquests elements musicals de la figura 4.18, l'aplicació reconeix com a negres els elements 1,2,3,6 i com a corxeres els elements 4,5,7,8,9 i 10.

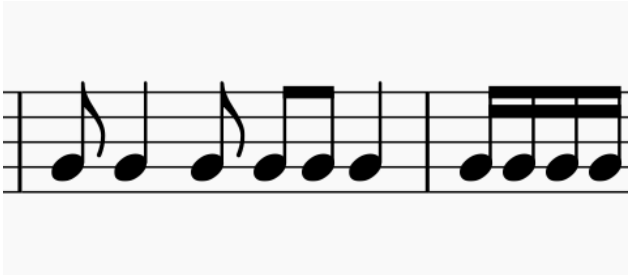


Figura 4.18: Reconeixement de corxeres II **Font:** Pròpia

4.9. Conversió de format

Un cop creats tots els objectes Nota es passa a la última etapa del procés de reconeixement: convertir els objectes Nota a notes reals de l'estàndard MIDI [7]. Per fer-ho, utilitzo la llibreria *MIDIUtil* de *Python*, 4.3.4.

El primer pas és crear un nou objecte MIDI i afegir-hi el tempo i un nom genèric. Per tots els objectes MIDI que genero defineixo un tempo de 140bpm.

A continuació miro tots els objectes Nota de tots els ritmes reconeguts (negres, corxeres, rodones i blanques), i en funció del ritme que sigui li assigno una durada diferent (rodones = 4 temps, blanques = 2 temps, negres = 1 temps i corxeres 0,5 temps). Recordeu que tota la informació bàsica referent a aspectes musicals està a l'apèndix A.

A l'últim pas afegeixo les notes, guardo el contingut de l'objecte MIDI a un arxiu MIDI i amb el sintetitzador PySynth [26] el converteixo a un fitxer auditiu amb format *".wav"* .

Les funcions que utilitzo de la llibreria MIDIUtil són:

```
#inicia l'objecte MIDI
midi = MIDIFile(1)
#afegeix el títol, "Track"
midi.addTrackName(track, time, "Track")
#afegeix el tempo, 140bpm
midi.addTempo(track, time, 140)
#per cada objecte Nota afegim una nota
midi.addNote(track,channel,freq,time,durada,volume)
```

L'aplicació s'acaba amb aquestes línies:

```
#obrim arxiu per ser escrit
arxiu_musica = open(partitura+'.mid', 'wb')
#passem l'objecte MIDI a l'arxiu MIDI
midi.writeFile(arxiu_musica)
#utilitzem el sintetitzador PySynth per transformar el fitxer MIDI a àudio
os.system("python3 "+readmidi+"/readmidi.py "
        +partitura+".mid 1 "+partitura+".wav")
```

Una altra opció de convertir els fitxers .mid a .wav és utilitzant el software Timidity [30], però des de l'entorn web PythonAnyWhere no disposo dels permisos suficients per instal · lar una aplicació amb la comanda "*sudo*"(super-usuari), per això he necessitat un sintetitzador que es pugui executar sense instal · lar-se.

4.10. Classes orientades a objectes

Per dur a terme les etapes de l'aplicació he cregut necessari crear dues classes d'objectes i funcions específiques. Aquestes classes són la classe Rectangle i la classe Nota.

4.10.1. Classe Rectangle

Aquesta classe disposa de sis funcions. Aquest mòdul retorna la ubicació de l'objecte rectangle i el dibuixa a la imatge *img*. Importa funcions de la llibreria *OpenCV* i de la llibreria *math*.

Els atributs que utilitza estan definits a la taula 4.3:

Taula 4.3: Definició dels atributs de la classe Rectangle

Atribut	Tipus	Significat
x	float	coordenada x en l'espai de coordenades (x,y)
y	float	coordenada y en l'espai de coordenades (x,y)
w	float	amplada
h	float	alçada
img	numpy.ndarray	imatge
color	tuple	color
thickness	int	gruix del contorn del rectangle a dibuixar

A l'apèndix E.3 hi ha la documentació completa de la classe Rectangle.

4.10.2. Classe Nota

Aquesta classe disposa d'una funció. Aquest mòdul no retorna res. Inicia un objecte Nota i calcula la freqüència de l'objecte Rectangle rect. importa funcions del mòdul Rectangle.

Els atributs que utilitza estan definits a la taula 4.4:

Taula 4.4: Definició dels atributs de la classe Nota

Atribut	Tipus	Significat
rect	objecte Rectangle	objecte Rectangle corresponent a un element musical
elem	str	nom de l'element musical
linia_rect	objecte Rectangle	objecte Rectangle corresponent a una línia de partitura
sostinguts	list	llista d'elements Nota referents a l'element musical sostingut
bemolls	list	llista d'elements Nota referents a l'element musical bemoll
nota	str	nom de la nota en notació anglesa
freq	int	freqüència de la nota en notació MIDI

A l'apèndix E.4 hi ha la documentació completa de la classe Nota.

CAPÍTOL 5

Servidor web

5.1. Introducció

Un objectiu inicial del projecte és crear una pàgina web accessible des de qualsevol interfície des d'on tothom i de forma lliure ,(sense la necessitat d'autentificar-se ni compartir dades personals), pugui utilitzar l'aplicació desenvolupada en aquest treball.
Per aconseguir-ho, he escollit un Framework per dissenyar pàgines web i, gràcies a un entorn de desenvolupament en línia, he aconseguit obrir a tothom aquest treball.

5.2. Frameworks més comuns

Un *framework* o entorn de treball, en el desenvolupament de programari és una estructura conceptual i tecnològica d'assistència definida, normalment, amb mòduls concrets de programari, que pot servir de base per a l'organització i desenvolupament de programari. Típicament, pot incloure suport de programes, biblioteques, i un llenguatge interpretat, entre d'altres eines, per així ajudar a desenvolupar i unir els diferents components d'un projecte.

Existeixen diferents entorns de desenvolupament web, a continuació explico breument els tres frameworks que he comparat per decidir quin utilitzar.

5.2.1. Web2py

Web2py [31] és un entorn de desenvolupament web de codi obert escrit en el llenguatge de programació *Python*. Web2py permet als desenvolupadors web programar contingut web dinàmic mitjançant Python. Està dissenyat per ajudar a reduir les tasques de desenvolupament web com ara desenvolupar formularis web des de zero.

Figura 5.1: Logo web2py **Font:** https://www.web2py.com

Web2py es va dissenyar originalment com una eina d'ensenyament amb èmfasi en la facilitat d'ús i implementació. Per tant, no té cap fitxer de configuració de nivell de projecte. El disseny de web2py estava inspirat en els frameworks de Ruby on Rails i Django. Igual que aquests marcs, web2py se centra en el desenvolupament ràpid, afavoreix l'enfocament de la convenció sobre la configuració i segueix un model d'arquitectura model-vista-controlador (MVC).

Web2py és el framework menys conegut d'aquests tres i és el que té menys suport d'usuaris. En comparació amb DJANGO l'aspecte social és una de les grans diferències. Això no vol dir que sigui un mal framework, ja que en termes de velocitat i senzillesa és el millor dels tres, sinó que és un nou entorn desenvolupat fa relativament pocs anys i poc utilitzat per la gran massa d'usuaris. Està pensat per a petites aplicacions o prototips web, però que no compleix els requisits de qualitat de la classe empresarial, ja que la complexitat de la resolució d'errors augmentarà de forma exponencial a causa de la manca de proves unitàries, d'informes d'errors correctes i precisos i d'un model dispers.

5.2.2. Django

Django [32] defineix com un entorn de desenvolupament web programat amb llenguatge *Python* d'alt nivell que afavoreix el desenvolupament ràpid i el disseny net i pragmàtic. Construït per desenvolupadors experimentats, s'ocupa de moltes de les molèsties del desenvolupament web, de manera que l'usuari es pugui centrar en escriure l'aplicació sense molts esforços. És de codi obert i gratuït. Els principals trets identificadors de Django són la seguretat, la rapidesa en la execució i la capacitat d'escalar un projecte.

Django és el framework més famós que utilitza Python. Això es pot comprovar amb l'anàlisi que he fet a la taula 5.1:

Taula 5.1: Comparació de la base social de DJANGO amb WEB2PY
Font: GitHub, Reddit i Stackoverflow

	web2py	**DJANGO**
Estrelles en el projecte a GitHub	1.588	34.001
Subscriptors a reddit.com	245	29.133
Preguntes a Stackoverflow	1.945	170.415

Django està pensat per tot tipus de projectes, des dels més senzills fins als més complexes. Per això hi ha tants usuaris que l'utilitzen, ja que inclús és una eina utilitzada per grans empreses gràcies.

Figura 5.2: Logo Django **Font:** https://www.djangoproject.com/

5.2.3. Wordpress

WordPress [33] és un sistema de gestió de continguts gratuït i de codi obert basat en els llenguatges de programació *PHP* i *MySQL*. Wordpress utilitza una arquitectura de connectors i un sistema de plantilles pel seu disseny. Actualment, WordPress és utilitzat per més del 30 % dels llocs web de tot internet. No cal dubtar doncs, que és el sistema de gestió de llocs web o blogging més popular en ús a la Xarxa, donant suport a més de 60 milions de llocs web [34].

WordPress va ser llançat el 27 de maig de 2003 pels seus fundadors, Matt Mullenweg i Mike Little. Està publicat sota la llicència GPLv2 (o posterior) [35].

Figura 5.3: Logo Wordpress **Font:** https://ca.wordpress.org/

La principal diferència amb els altres dos frameworks presentats és que amb WordPress es pot crear un lloc web en qüestió de segons i sense tenir cap coneixement tècnic, ja que utilitza la interfície gràfica per personalitzar l'entorn web.

5.3. Framework escollit

L'entorn de treball que he escollit pel desenvolupament web és **Django**. He optat per aquest framework per diferents motius:

- Volia utilitzar un entorn nou. A l'assignatura d'Aplicacions i Serveis vam

aprendre a utilitzar Web2py i Django era desconegut per a mi.

- Wordpress no el podia utilitzar ja que és un framework que no utilitza Python.
- Django és un framework que treballa amb Python: el mateix llenguatge de programació que he utilitzat per escriure el programa de reconeixement. Personalment un llenguatge d'alt nivell amb una sintaxi senzilla i entenedora.

5.4. Entorn i allotjament web

Per allotjar el servidor Django a Internet i fer-lo accessible a tothom he utilitzat l'entorn *PythonAnywhere* [36], un entorn de desenvolupament integrat en línia i servei d'allotjament web basat en el llenguatge de programació *Python*. Proporciona accés amb el navegador a interfícies de la línia de comandaments de *Python* i *Bash* basades en el servidor, juntament amb un editor de codi amb ressaltat de sintaxi. També disposa d'una interfície gràfica des d'on es poden llegir i modificar tots els fitxers existents al compte.

Figura 5.4: Logo PythonAnyWhere **Font:** https://www.pythonanywhere.com/

He creat un usuari amb un compte gratuït, el que em limita a 512MB la capacitat màxima d'emmagatzematge del servidor, i una caducitat de l'entorn al cap de 3 mesos. L'URL de la web és "*joanbruch.pythonanywhere.com*" [37].

5.4.1. Seguretat

La seguretat és un element molt important a tenir en compte alhora de treballar amb pàgines web d'accés a Internet. Entre d'altres motius, vaig escollir PythonAnyWhere com a domini del servidor perquè proporciona el certificat SSL que

permet utilitzar el protocol HTTPS [38].

SSL encripta la comunicació entre el punt A i el punt B (el servidor web i el navegador). Aquest xifrat és important per una raó específica: impedeix que algú pugui ser capaç d'interceptar aquest trànsit, conegut com un atac *"man in the middle"*. SSL és especialment important per llocs web de comerç electrònic i qualsevol lloc web que accepta enviaments de formularis amb les dades de l'usuari sensibles o informació d'identificació personal. En el meu cas, les dades que s'omplin al el formulari d'opinions només les veurem l'usuari i jo.
Tot i això, el protocol HTTPS [39] no fa res per protegir el lloc web contra qualsevol atac maliciós, o obligar a que es deixi de distribuir malware. He utilitzat l'eina que proporciona SSL Labs per explorar el servidor web SSL. Aquesta pàgina web [40] proporciona un anàlisi en profunditat de la pàgina web, incloent-hi el dia de caducitat del certificat SSL, la qualificació general, el Xifrat, la versió SSL / TLS, la informació del protocol,... A l'apèndix F mostro els resultats detallats de la inspecció. A la figura 5.5 mostro el resum de l'anàlisi:

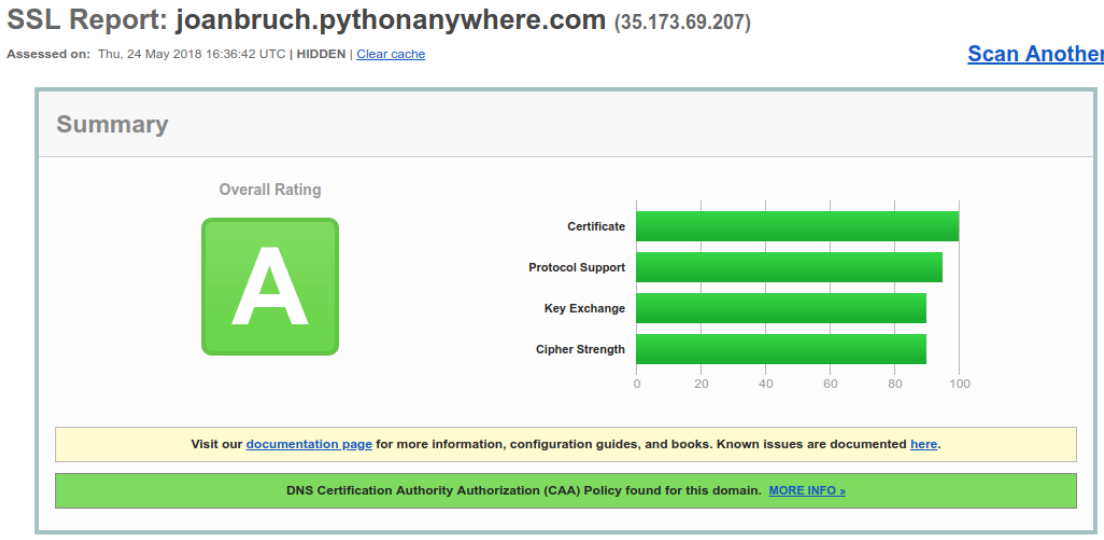


Figura 5.5: Resum de l'anàlisi de seguretat **Font:** https://www.ssllabs.com

Podem comprovar que el resultat és molt favorable, amb una nota **A**.

Permisos

Existeixen dos perfils d'usuari diferent: l'administrador propietari de la web i els clients. L'administrador té accés (amb usuari i contrasenya) a modificar el contingut total de la pàgina web i del servidor. A continuació explico els permisos que té el perfil dels clients segons les accions possibles:

- **Llegir la pàgina web:** de forma lliure i sense restriccions
- **Executar el programa:** de forma lliure i sense restriccions
- **Enviar formulari:** de forma lliure i sense restriccions. Per seguretat he incorporat l'eina reCaptcha [41] de Google. És una eina dissenyada per prevenir atacs cibernètics amb els que un bot sobrecarrega la pàgina web amb enviaments de formulari continuats.

No és necessari iniciar sessió per accedir a la pàgina web ni per executar el programa Music Scanner. Tothom és lliure de veure tots els continguts de la pàgina web, inclosos els fitxers que es troben al directori */media/documents* on es guarden tots els fitxers que s'han carregat a la pàgina web (imatge original, imatge amb els patrons dibuixats, fitxer MIDI i fitxer d'àudio). Tot i així, com que cada usuari només coneix aquells fitxers que ell mateix ha carregat al servidor, resulta difícil que un usuari pugui veure els fitxers que han carregat els altres usuaris.

5.5. Interfície d'Usuari

Les interfícies d'usuari són les responsables de la interacció entre els homes i les màquines.
En aquest treball, la interfície d'usuari és la web *joanbruch.pythonanywhere.com* . Segons s'explica en aquest document de l'Institut Obert de Catalunya [42], és molt important oferir una bona usabilitat perquè una pàgina web amb una bona usabilitat tindrà molt més èxit que una altra encara que tinguin continguts o informacions

similars, però una usabilitat no tan bona.

Una interfície web és usable quan l'usuari pot aconseguir un objectiu concret. Com menys temps passi per aconseguir l'objectiu en qüestió, més usable és. És la qualitat d'una pàgina web o d'un programa informàtic de ser fàcil d'usar i de tenir en compte aspectes com ara la llegibilitat dels textos, la rapidesa de la baixada d'informació, la manejabilitat i la capacitat de satisfer les necessitats de l'usuari.

5.5.1. Avaluació de la interfície

Per avaluar una interfície d'usuari es poden utilitzar moltes tècniques diferents. Aquestes tècniques aconsegueixen resultats qualitatius i quantitatius.
Els resultats qualitatius (resultat d'enquestes, de valoracions subjectives, de preguntes amb resposta oberta, opinions del producte, etc) no els podem quantificar però són un aspecte important com els altres, els resultats quantitatius, que els analitzem recollint informació objectiva com pot ser:

- Temps dedicat a cada interfície abans d'escollir una opció
- Temps o nombre de vegades d'utilització de l'ajuda o documentació
- Freqüència d'ús de les opcions
- Nombre d'errors en l'ús de les interfícies per usuari o cada cert temps
- Nombre d'accions completades en un temps determinat
- Comparació de temps utilitzat en segones o terceres visites a una mateixa interfície
- Nombre de suggeriments o queixes del producte

Per avaluar la interfície s'ha de realitzar la prova amb usuaris de diferents nivells d'edat, de coneixement i d'habilitats, per assegurar que tothom és capaç d'utilitzar la interfície perfectament.

5.6. Manual d'usuari

En aquesta secció explico els passos a seguir per escanejar una partitura des de la pàgina web *joanbruch.pythonanywhere.com* [37].

A la part superior esquerra de la pàgina principal hi apareix un menú que és un accés directe a les tres seccions de la pàgina principal: *El projecte, Exemples d'èxit i Contacte*. Si la pantalla des d'on es visualitza la web és petita, el menú apareixerà a la cantonada superior dreta amb una icona de tres barres horitzontals.

Figura 5.6: Pàgina principal **Font:** joanbruch.pythonanywhere.com

A la secció *El Projecte* explico breument de què es tracta la pàgina web i el projecte. També explico els principals requisits que ha de tenir tota partitura per ser escanejada amb un alt tant per cent d'èxit en el reconeixement.

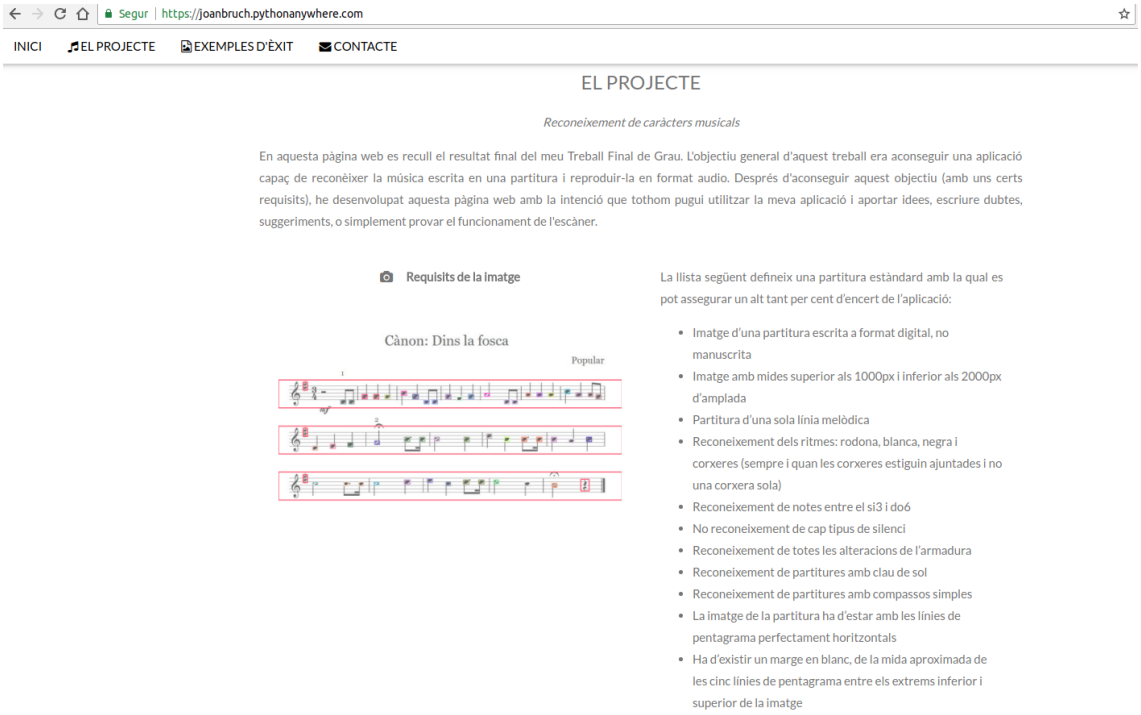


Figura 5.7: El projecte **Font:** joanbruch.pythonanywhere.com

Dins la mateixa secció del projecte hi ha un botó per seleccionar el fitxer de la partitura i executar l'aplicació *"Music Scanner"*.

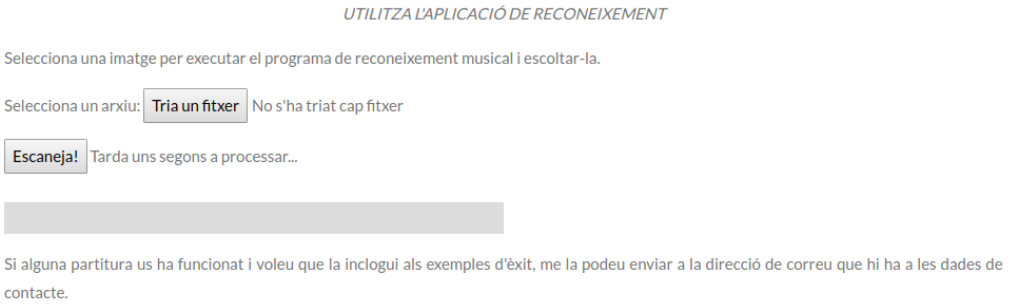


Figura 5.8: Seleccionar fitxer **Font:** joanbruch.pythonanywhere.com

La secció següent és la d'*Exemples d'èxit*. Aquí hi ha un breu recull de partitures que l'aplicació reconeix perfectament o amb un alt tant per cent d'èxit.

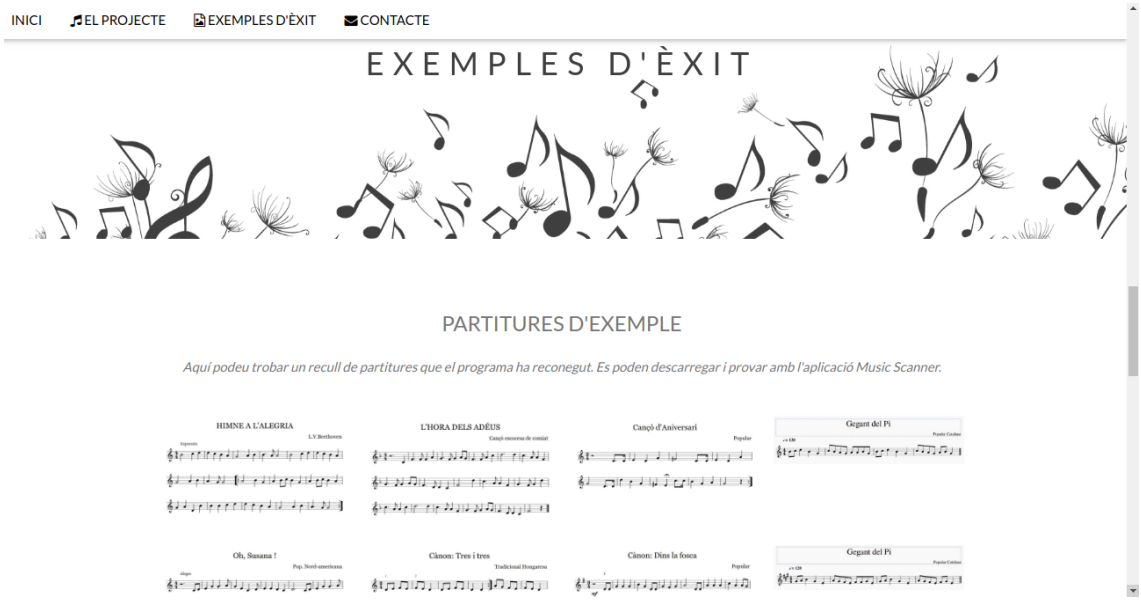


Figura 5.9: Exemples d'èxit **Font:** joanbruch.pythonanywhere.com

Per accedir a la llista completa de partitures que el programa reconeix a la perfecció o amb un tant per cent elevat cal clicar el botó *"Més exemples"*.

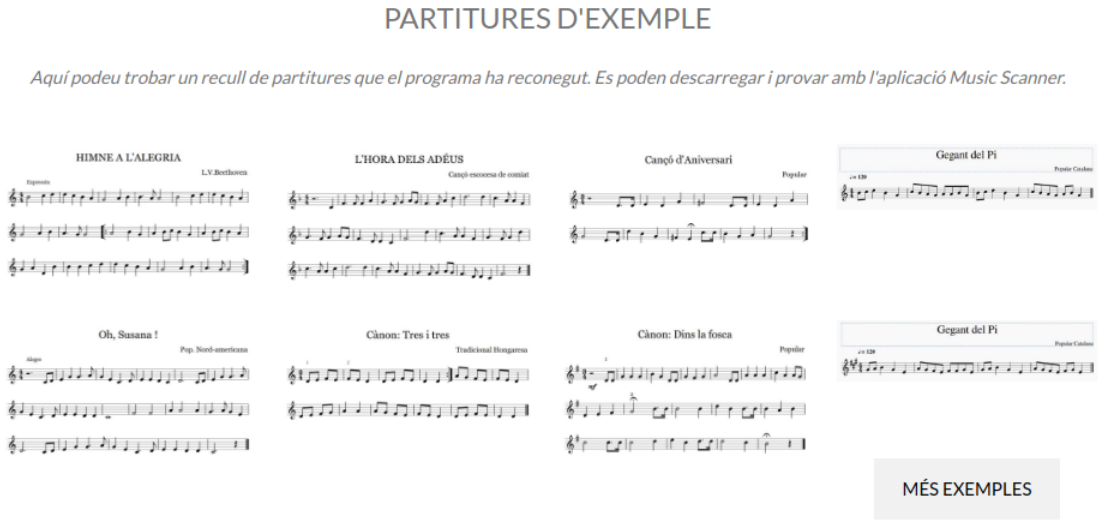


Figura 5.10: Botó "Més Exemples"**Font:** joanbruch.pythonanywhere.com

Finalment, a la part inferior de la pàgina principal hi ha la secció de *Contacte*. Es mostra una adreça de correu on es poden enviar imatges de partitures que s'han reconegut o que no, i disposa d'un formulari amb els camps *Nom*, *Email* i *Missatge* per escriure les opinions i comentaris. Tot el que s'escrigui s'envia a l'adreça de

correu *musicscanner.jbruch@gmail.com* .

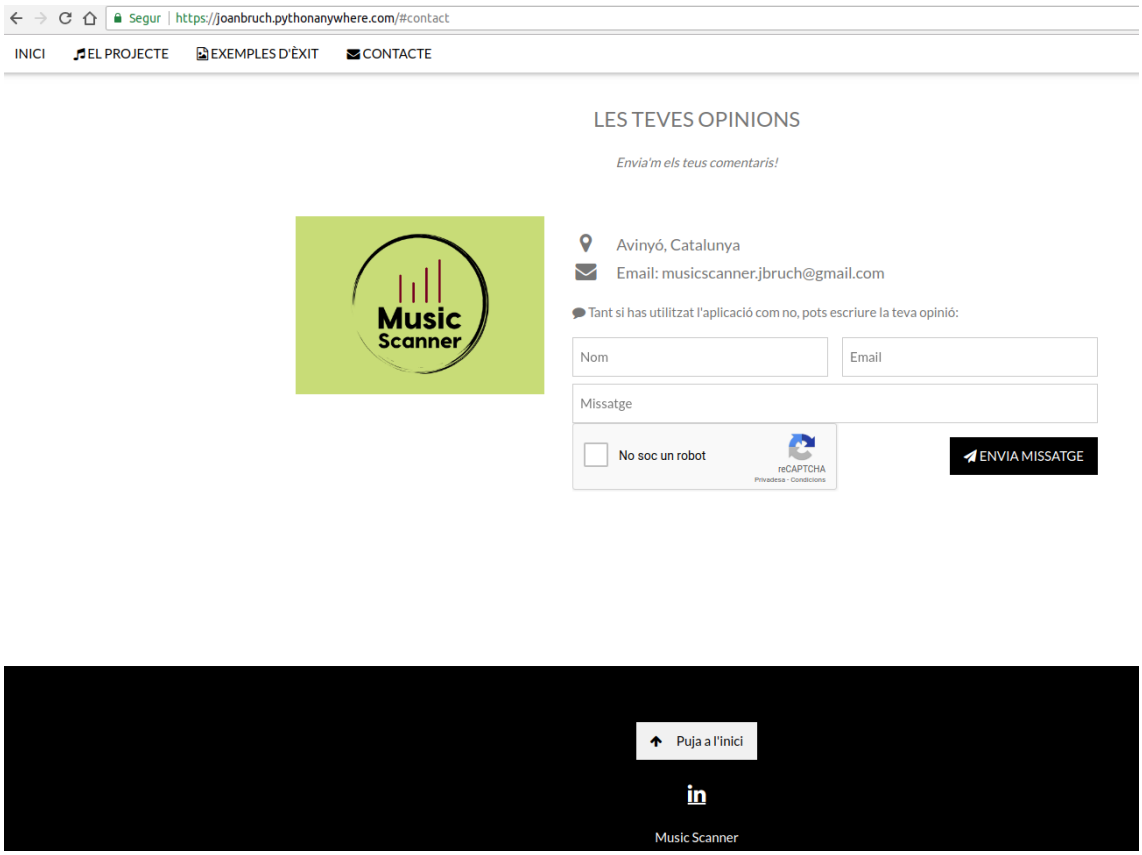


Figura 5.11: Contacte **Font:** joanbruch.pythonanywhere.com

Com he explicat a la captura 5.10, si es clica el botó *"MÉS EXEMPLES"* s'obre una pàgina web amb el llistat total de partitures que es poden escanejar amb un alt percentatge d'encert. Si cliquem damunt de qualsevol cançó s'obrirà amb gran i es podrà descarregar per la posterior utilització en l'aplicació de reconeixement.
Si cliquem en el botó *"PRINCIPAL"* del menú superior dret, tornarem a anar a la pàgina principal.

Figura 5.12: Partitures de mostra **Font:** joanbruch.pythonanywhere.com

Si s'intenta escanejar una partitura però el programa pateix un error inesperat, saltarà a la pàgina web següent indicant que hi ha hagut un error en el reconeixement de la partitura que s'ha carregat al programa, i aportant un llistat de solucions.

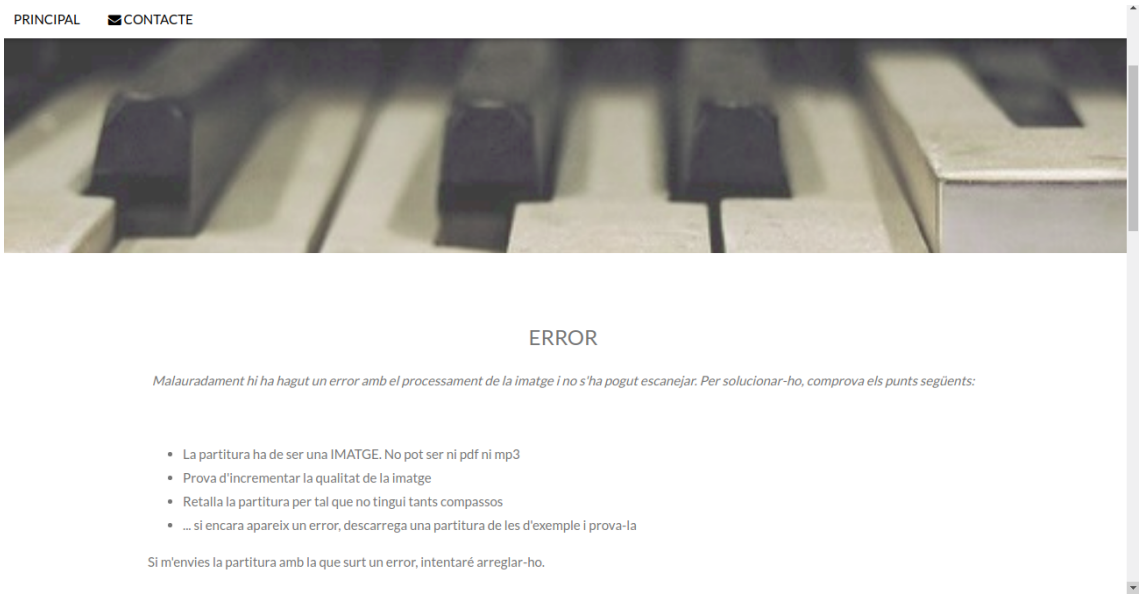


Figura 5.13: Error en el reconeixement **Font:** joanbruch.pythonanywhere.com

5.6.1. Sistemes Operatius i navegadors disponibles

He comprovat que la pàgina web i l'aplicació *Music Scanner* es poden executar a tots els Sistemes Operatius següents:

- Windows (versió Windows Vista o superior)
- Android (versió 5 o superior)
- Ubuntu (versió 16.04)

Llista de navegadors comprovats que permeten l'accés a totes les accions de la pàgina web:

- Google Chrome
- Mozilla Firefox

Amb els dispositius de la marca Apple es pot visualitzar la pàgina web i executar el programa però no permeten la reproducció del fitxer àudio resultant.

CAPÍTOL 6

Resultats finals del programa

6.1. Introducció

En aquest apartat s'exposen els resultats de les proves realitzades a l'aplicació amb diferents partitures. Com que l'algorisme ha anat rebent modificacions per tal de millorar-lo, mostro els resultats del test de la última versió (amb data 24 de maig).

He generat partitures de prova i també he buscat partitures per la xarxa per realitzar els tests propis. Per altra banda, les partitures dels usuaris que han utilitzat l'aplicació em serveixen com a element per incloure en els tests.

A l'apèndix D hi apareixen les imatges de les partitures utilitzades.

6.2. Test propis

Per avaluar els resultats he tingut en compte dos tipus diferents de reconeixement:

- Reconeixement correcte del ritme i alteracions
- Reconeixement correcte de les notes

A continuació mostro un esquema d'avaluació amb els diferents pesos que he definit per cada element:

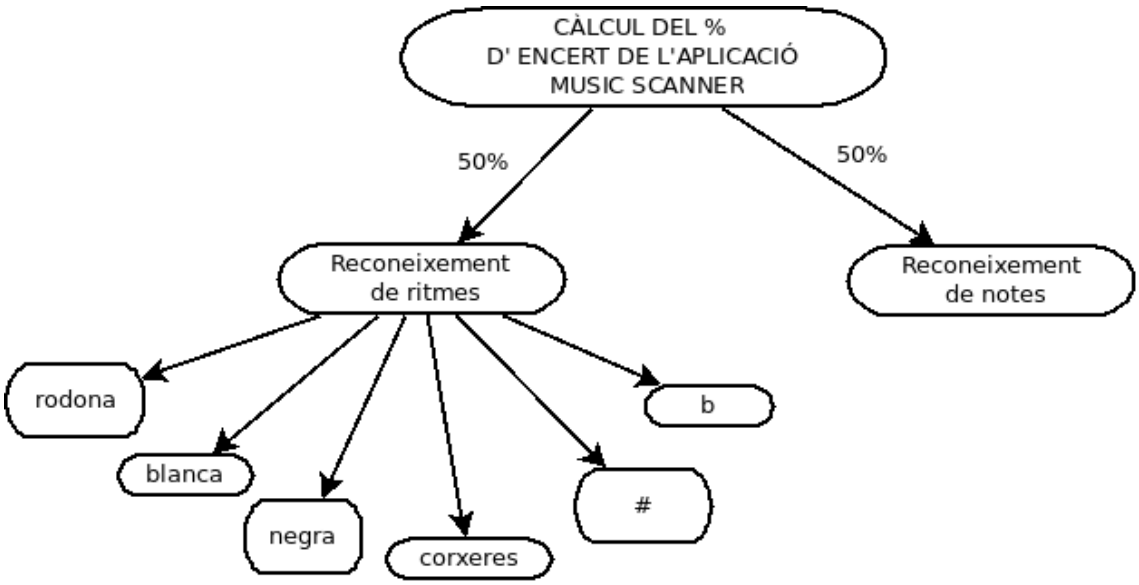


Figura 6.1: Esquema del sistema d'avaluació **Font:** Pròpia.

Seguint els percentatges establerts a l'esquema d'avaluació, a la partitura 6.2 hi ha 26 patrons per ser reconeguts (25 notes + 1 alteració), i 25 notes que cal identificar la seva altura en el pentagrama.

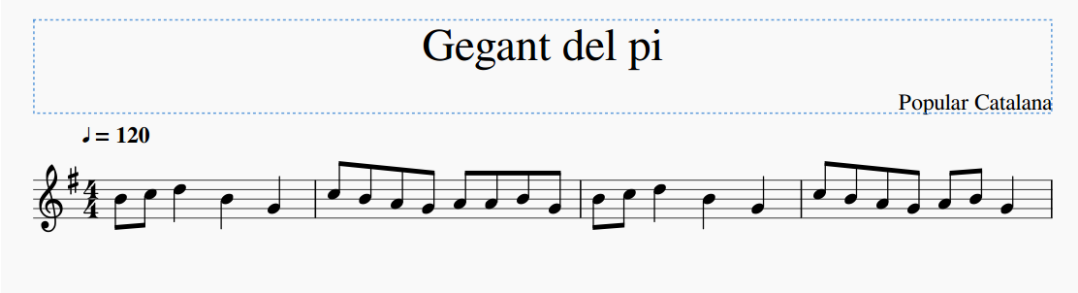


Figura 6.2: Partitura d'exemple pels tests **Font:** Pròpia.

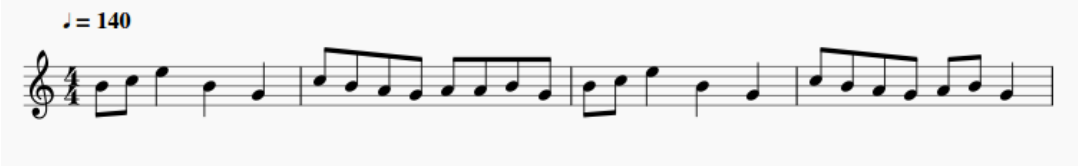


Figura 6.3: Resultat del Music Scanner per la partitura anterior **Font:** Pròpia.

Els resultats del reconeixement d'aquesta partitura són:

- Ritmes i alteracions reconeguts: 26/26

- Notes reconegudes: 23/25

El resultat final del reconeixement és:

$$Reconeixement[\,\%] = \frac{26}{26} * 50 + \frac{23}{25} * 50 = 96\,\% \tag{6.1}$$

6.2.1. Les alteracions: bemolls i sostinguts

Per reconèixer bé aquestes alteracions cal tenir en compte que les podem trobar en diferents llocs de la partitura. Aquestes posicions són:

- A l'armadura
- Al costat esquerra d'una nota

I cada posició té un significat diferent:

- Les alteracions a l'armadura signifiquen que totes les notes de la partitura les quals tinguin alteracions escrites a l'armadura seran alterades (incrementant o reduint mig to) en funció de si l'alteració és un sostingut o un bemoll.
- Les alteracions situades al costat esquerra d'una nota només tenen efecte en aquella precisa nota i totes les notes iguals del mateix compàs.

Tal com està definit en l'estàndard, l'aplicació només té en compte partitures amb alteracions en l'armadura. Per tant, totes les partitures que tinguin alteracions fora de l'armadura no s'assegura que es reconeguin ni que no.
Per dur a terme tests amb les alteracions m'he creat una partitura amb el programa MuseScore [20] d'una cançó fàcil i curta per ser reconeguda amb rapidesa pel meu programa i així utilitzar-la amb diferents alteracions, que és el que avaluo en aquest apartat.

6.3. Partitures d'usuaris de la web

Des que vaig compartir la pàgina web, molts usuaris han provat el funcionament de l'aplicació de reconeixement. No he pogut comprovar quants tests han realitzat els usuaris perquè el domini *pythonanywhere* em permet una memòria de 512MB que es van omplir al cap de poc (tenint en compte que bona part de l'espai d'emmagatzematge l'ocupa el servidor), i algunes partitures no s'han guardat. M'he descarregat els arxius per buidar la memòria, i tinc un total de 38 partitures que la gent ha carregat al servidor. D'aquestes, la gran majoria no han passat l'escàner perquè són partitures amb baixa qualitat, o no compleixen els requisits bàsics de l'estàndard (partitures de més d'una línia melòdica, partitures amb clau de fa, etc). Aquestes partitures no les he avaluat de forma tan acurada com les dels tests propis, sinó que les he avaluat de la següent forma:

- Partitures amb un bon reconeixement: 2
- Partitures amb un mal reconeixement: 16
- Partitures que el programa ha retornat error: 19

6.4. Avaluació dels resultats

Per avaluar la capacitat de reconeixement de l'aplicació he testejat un total de 31 imatges de partitures diferents. Totes les imatges es troben a l'apèndix.
D'aquestes 31 imatges de partitures, 14 són imatges capturades d'Internet, 12 són partitures creades per mi amb l'editor de partitures MuseScore [20], i 5 són partitures escanejades del llibre "*Aprèn jugant amb la flauta travessera 1*" [1].
Els resultats són els que es mostren a la taula 6.1:

Taula 6.1: Resultat dels tests propis

Test núm	**nom_partitura**	**Encert Patrons**	**Total Patrons**	**Encert Notes**	**Total Notes**	**ENCERT TOTAL [%]**
1	gegant_DOM.png	25	25	25	25	100,00
2	gegant_REM.png	27	27	25	25	100,00
3	gegant_MIM.png	29	29	25	25	100,00
4	gegant_FAM.png	26	26	25	25	100,00
5	gegant_SOLM.png	26	26	23	25	96,00
6	gegant_LAM.png	24	28	25	25	92,86
7	gegant_SIM.png	26	30	25	25	93,33
8	gegant_DobM.png	27	32	19	25	80,19
9	gegant_RebM.png	30	30	23	25	96,00
10	gegant_MibM_agut.png	28	28	0	25	50,00
11	gegant_MibM_greu.png	28	28	25	25	100,00
12	gegant_SOLbM.png	31	31	25	25	100,00
13	dinslafosca.jpg	52	52	49	49	100,00
14	moltesfelicitats.jpg	27	27	25	25	100,00
15	tresitres.jpg	39	39	39	39	100,00
16	elmati.jpg	101	103	83	96	92,26
17	himnealegria.jpg	57	57	57	57	100,00
18	horadelsadeus.jpg	59	59	59	59	100,00
19	ohsusana.jpg	55	55	55	55	100,00
20	bobdylan.jpg	80	80	80	80	100,00
21	jinglebells.png	89	149	0	141	29,87
22	scarboroughfair.jpg	43	43	29	43	83,72
23	titanic.jpg	76	87	58	81	79,48
24	baixantfontdelgat.jpg	28	32	32	32	93,75
25	boigpertu.jpg	67	67	66	67	99,25
26	santanit.jpg	46	50	34	50	80,00
27	quanlesoques.jpg	27	29	14	29	70,69
28	undosra.jpg	29	34	0	34	42,65
29	picapedrer.jpg	24	25	0	25	48,00
30	aravenadal.jpg	16	21	0	21	38,10
31	rellotge.jpg	45	46	14	46	64,13

Seguint els criteris d'avaluació, la mitjana d'encert és del 84,85 % .

6.4.1. Conclusions

Detectar les línies del pentagrama correctament és l'acció principal del reconeixement. Un pentagrama mal detectat provoca que l'altura de les notes no sigui la correcta, encara que la resta de patrons els detecti perfectament. A la següent imatge mostro una partitura amb el pentagrama detectat malament. Per arreglar això, s'han de modificar els valors que defineixen l'escalat mínim i màxim del patró del pentagrama en funció de les mides de la imatge.

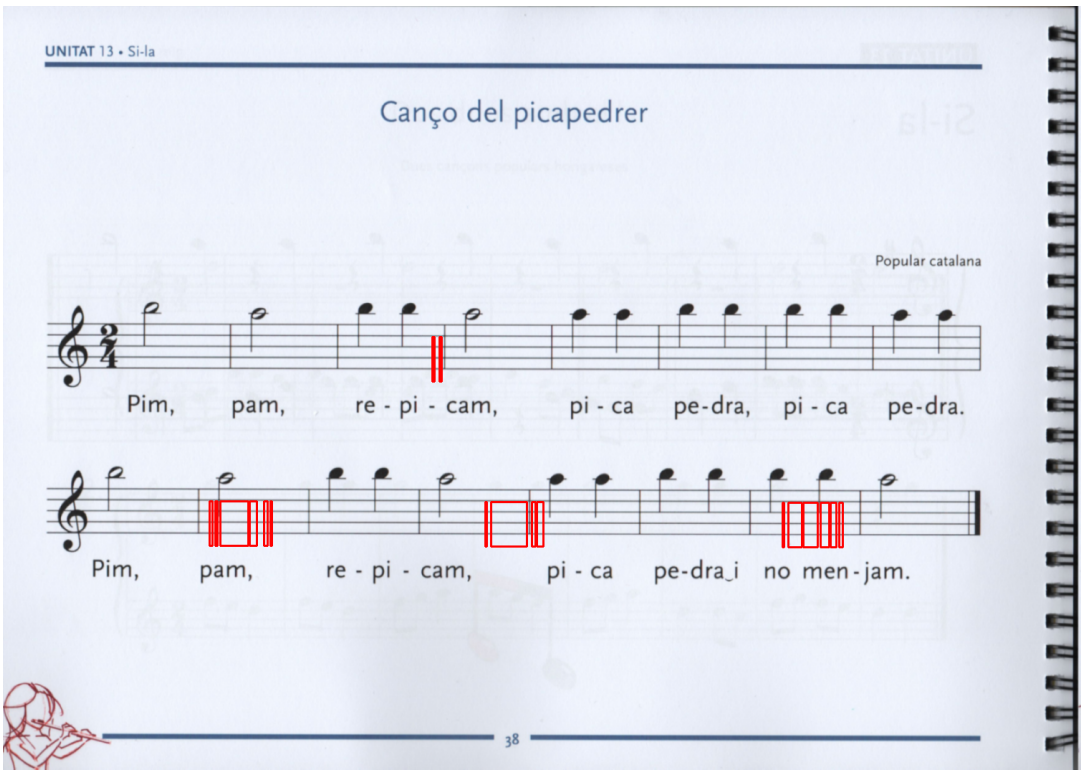


Figura 6.4: Error en el reconeixement del pentagrama **Font:** Pròpia.

Per aconseguir que algunes partitures es reconeguessin correctament he hagut de canviar manualment les mides de la imatge (augmentar-les o reduir-les), ja que per valors superiors a 2000px d'amplada i per valors inferiors a 1000px d'amplada no reconeixia bé perquè el mètode de reducció/ampliació de les imatges no és prou eficaç per valors extrems i els paràmetres de l'escalat dels patrons són fixes per qualsevol mida de partitura.

Això és un punt que explicaré al capítol de Línies Futures 8, la millora del programa perquè reconegui un major ventall de mides de partitures modificant automàticament les mides de la imatge de la partitura segons els millors resultats. (recordar que ara només modifica les imatges dels patrons).

També hi ha més possibilitat que reconegui les partitures capturades d'Internet que no pas les escanejades des d'un llibre, ja que la qualitat de la imatge és més nítida les imatges capturades d'Internet que les que he escanejat.

Observant els resultats de la taula 6.1, considero que una partitura s'ha escanejat de forma entenedora quan l'encert supera el 85 %. Per tant, de les 31 partitures testejades, 20 han superat el llindar del 85 % d'èxit, un 64,52 % de les partitures testejades.

CAPÍTOL 7

Conclusions

Desenvolupar l' aplicació de reconeixement de caràcters musicals *Music Scanner* m'ha suposat un repte molt important; no només pel fet de la dificultat en sí mateixa del programa de reconeixement, sinó que la construcció d'una interfície web d'accés lliure i gratuït ha afegit una feina extra al treball.

Puc afirmar que he complert l'objectiu general i gran part dels objectius específics. Els objectius específics que no he pogut aconseguir estan explicats al capítol 8.

El treball consta d'un sistema OMR que es pot executar des de la pàgina web joanbruch.pythonanywhere.com . Seguint els requisits de l'estàndard es pot aconseguir un percentatge d'encert superior al 85 %.

Per realitzar el treball he hagut d'aprendre noves llibreries de Python desconegudes fins ara per mi, com OpenCV, NumPy i MIDIUtil. També he hagut de construir un lloc web amb unes eines desconegudes com DJANGO i PythonAnyWhere.
A l'hora d'escriure la memòria he optat per l'eina Latex, tot i que he hagut de fer un esforç d'entesa de l'eina degut a què no l'havia utilitzat fins ara, he arribat a la conclusió que és una eina molt útil per la realització de documents.

El resultat final de l'aplicació de reconeixement està bastant limitat amb el nombre d'elements musicals que pot reconèixer i les característiques que ha de tenir la

partitura. Vaig necessitar molts dies per aconseguir una idea inicial ferma de com escriure el codi OMR i m'ha perjudicat al final, perquè m'ha impedit millorar el programa incorporant millores en les etapes de preprocessament de la imatge i del reconeixement que haurien augmentat la capacitat de detecció de l'aplicació.

Estic molt satisfet amb el resultat final de la interfície web. És una interfície senzilla i segura que permet executar l'aplicació de reconeixement des de qualsevol dispositiu dels esmentats al punt 5.6.1.

Resumint, estic satisfet del resultat de tot el conjunt desenvolupat durant aquests mesos: redacció de la memòria, programació del sistema OMR i elaboració d'un servidor.

CAPÍTOL 8

Línies futures

Malgrat haver complert la gran part dels objectius, al mateix temps que anava desenvolupant l'aplicació i sobretot a l'hora de realitzar els diferents tests, m'han sorgit un seguit d'idees complementàries per millorar el treball.

8.1. Millores en el programa de reconeixement

A continuació explico mètodes i funcions que es poden realitzar al programa OMR per incrementar-ne la capacitat de detecció i processament de la música escrita.

8.1.1. Incorporar correcció de l'orientació

Un requisit de l'estàndard és que les partitures han d'estar perfectament horitzontals, ja que si les línies del pentagrama estan tortes, l'altura de les notes no es reconeix correctament. Per corregir l'orientació utilitzaria la funció "*cv2.GetRotationMatrix2D(center, angle, scale, mapMatrix)*"de la llibreria OpenCV [3]. Aquesta funció gira la imatge segons el valor *angle* que se li passa com a paràmetre.

8.1.2. Més ritmes per reconèixer

Afegir més patrons per reconèixer i augmentar la llista de possibilitats musicals que l'aplicació és capaç de processar. També augmentar la llista de patrons amb diferents mides de patrons de cada element musical.

8.1.3. Afegir els silencis musicals

Com es pot veure a les imatges resultants del programa OMR, els silencis de negra els reconeix i els dibuixa un rectangle al voltant d'ells. El problema és a l'hora de transformar els objectes nota a elements midi utilitzant la llibreria MIDIUtil [5]. No he trobat la manera d'escriure un silenci amb aquesta llibreria. Caldria investigar molt més a fons o buscar una llibreria diferent a aquesta, però amb la mateixa capacitat de crear un fitxer MIDI i que acceptés els silencis.

8.1.4. Més amplitud de notes

Actualment el programa reconeix notes que van des del si3 fins al do6 (i tot i així, les notes dels extrems agut i greu les diferencia amb dificultats). Caldria millorar l'algorisme per assegurar que les reconeix amb un millor encert i augmentar la llista de notes que pugui reconèixer.

8.1.5. Diferents claus

El programa és capaç de llegir partitures amb clau de Sol (la majoria de partitures estan escrites en aquesta clau). Però com explico a l'apèndix A, existeixen altres claus en la música. S'hauria d'afegir patrons de claus diferents i un cop el programa hagi reconegut quina clau és, modificar l'algorisme que determina l'alçada de les notes per tal que segons la clau de la partitura, escrigui unes notes o unes altres.

8.1.6. Permetre el reconeixement de música escrita a mà alçada

Aquest apartat és el més complex dels que he explicat en aquest capítol. Tenint en compte que ningú escriu exactament igual (alguns dibuixen el cap de les notes més ovalats, altres més gruixuts, altres dibuixen la plica de les notes més rectes o més tortes...) 8.1, caldria utilitzar una gran base de dades de patrons a mà alçada i contemplar totes aquestes possibilitats. Això afectaria la velocitat de processament

del programa, ja que com més patrons hagi de comparar, més temps tardarà a executar-se completament.

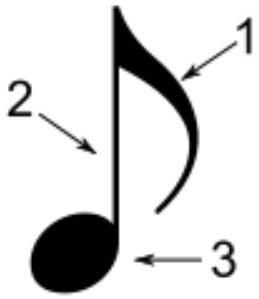


Figura 8.1: Parts d'una nota: 1-Corxet, 2-Plica, 3-Cap
Font: https://ca.wikipedia.org/wiki/Plica

8.2. Millora de la interfície web

Per fer més accessible a tots els perfils de la societat, caldria millorar la usabilitat i accessibilitat de la interfície web:

- Incorporar la internacionalització (possibilitat d'idiomes diferents).
- Realitzar tests d'usabilitat a persones de diferents nivells de coneixement informàtic i capacitats per descobrir els punts febles de la web i saber què cal millorar.

8.2.1. Afegir més personalització per l'usuari

El *tempo* és un concepte molt important en la música, ja que no totes les cançons es toquen igual de ràpides. Actualment l'aplicació genera l'àudio a un tempo de 140bpm genèric per totes les partitures processades, però es podria crear una entrada a la web perquè l'usuari pogués definir aquest valor.

De la mateixa manera, també es podria crear un desplegable amb un llistat d'instruments diferents i que l'usuari pogués triar amb quin instrument vol que soni l'àudio resul-

tant. Per defecte és el piano, però s'investigaria amb la llibreria MIDIUtil o amb altres llibreries de transformació a fitxers MIDI per fer-ho possible.

8.3. Crear una aplicació per a dispositius mòbils

Aquest és un dels objectius inicials que no he pogut assolir: desenvolupar una aplicació per a dispositius mòbils.
Una interfície d'usuari per a dispositius mòbils permet els usuaris interactuar amb les funcionalitats i característiques dels dispositius mòbils, com per exemple els telèfons intel · ligents i les tauletes tàctils. En el meu cas, interactuarien amb les càmeres del dispositiu mòbil o bé amb fitxers interns (imatges de partitures) ja existents en els terminals.

Principalment, hi ha tres maneres de desenvolupar interfícies d'usuari per a dispositius mòbils: aplicacions web per a mòbils, aplicacions natives i aplicacions híbrides.

Les aplicacions híbrides es dissenyen amb la mateixa tecnologia que les aplicacions web per a mòbils –HTML, CSS i JavaScript– però s'executen encastades en navegadors locals. Així, es dissenyen com a aplicacions multi-plataforma però posteriorment s'empaqueten específicament per a cada dispositiu. Aquestes aplicacions tenen accés a les funcionalitats extres dels dispositius mitjançant interfícies estàndard que adapten les especificitats de cada dispositiu.

Amb l'eina *Apache Cordova* és un entorn de desenvolupament d'aplicacions híbrides per a mòbils que permet dissenyar aplicacions amb tecnologia web (utilitzant els fitxers HTML i CSS guardats en el servidor de *PythonAnyWhere*) i empaquetar-los per a múltiples dispositius mòbils (Apple iOS, Firefox OS, Google Android, Microsoft Windows Phone, Ubuntu Touch, etc)

BIBLIOGRAFIA

[1] M. Lorenz, *Aprèn jugant amb la flauta travessera 1.* Amalgama, 2012, vol. 1.

[2] I. F. Ana Rebelo, "Optical music recognition: state-of-the-art and open issues," *International Journal of Multimedia Information Retrieval*, vol. 1, no. 3, pp. 173–190, Octubre 2012.

[3] Llibreria opencv. [En línia]. Disponible: https://opencv.org/ [Data de consulta: Març 2018]

[4] Paquet numpy. [En línia]. Disponible: http://www.numpy.org/ [Data de consulta: Març 2018]

[5] Llibreria midiutil. [En línia]. Disponible: https://pypi.org/project/MIDIUtil/ [Data de consulta: Abril 2018]

[6] A. Candelaria, *Teoria de la Musica, Nivel 1.* Smashwords Edition, 2015, vol. 1.

[7] Estàndard midi. [En línia]. Disponible: https://learn.sparkfun.com/tutorials/midi-tutorial/the-midi-standard [Data de consulta: Maig 2018]

[8] Estàndard de notació de partitures en format digital. [En línia]. Disponible: https://www.musicxml.com/ [Data de consulta: Abril 2018]

[9] Aplicació omr privada: Musitek. [En línia]. Disponible: http://www.musitek.com/ [Data de consulta: Maig 2018]

[10] Aplicació omr privada: Photoscore. [En línia]. Disponible: http://www.neuratron.com/photoscore.htm [Data de consulta: Maig 2018]

[11] Aplicació omr privada: Sharpeye. [En línia]. Disponible: http://www.music-scanning.com/sharpeye.html [Data de consulta: Maig 2018]

[12] Aplicació omr privada: Capella. [En línia]. Disponible: https://www.capella-software.com/us/index.cfm/products/capella-scan/info-capella-scan/ [Data de consulta: Maig 2018]

[13] Aplicació omr privada: Playscore. [En línia]. Disponible: http://www.playscore.co/ [Data de consulta: Maig 2018]

[14] Aplicació omr privada: Pdftomusic. [En línia]. Disponible: http://www.myriad-online.com/en/products/pdftomusicpro.htm [Data de consulta: Maig 2018]

[15] Aplicació omr lliure: Audiveris. [En línia]. Disponible: https://github.com/audiveris [Data de consulta: Maig 2018]

[16] Aplicació omr lliure: Gamera. [En línia]. Disponible: http://gamera.informatik.hsnr.de/ [Data de consulta: Maig 2018]

[17] Llindar d'otsu. [En línia]. Disponible: http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html [Data de consulta: Maig 2018]

[18] Algorisme knn. [En línia]. Disponible: https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/ [Data de consulta: Març 2018]

[19] Tècnica de l'associació de patrons. [En línia]. Disponible: https://docs.opencv.org/2.4.13.4/doc/tutorials/imgproc/histograms/template_matching/te
[Data de consulta: Març 2018]

[20] Programa per crear/editar partitures muse score. [En línia]. Disponible: https://musescore.org/ca [Data de consulta: Febrer 2018]

[21] Programa per crear/editar partitures: Finale. [En línia]. Disponible: https://www.finalemusic.com/ [Data de consulta: Maig 2018]

[22] Programa per crear/editar partitures: Sibelius. [En línia]. Disponible: http://www.avid.com/sibelius [Data de consulta: Sibelius 2018]

[23] Programa per crear/editar partitures: Lilypond. [En línia]. Disponible: http://lilypond.org/ [Data de consulta: Maig 2018]

[24] Llicència bsd. [En línia]. Disponible: http://www.linfo.org/bsdlicense.html [Data de consulta: Març 2018]

[25] Mòdul sys. [En línia]. Disponible: https://docs.python.org/2/library/sys.html [Data de consulta: Març 2018]

[26] Sintetitzador musical escrit amb llenguatge python. [En línia]. Disponible: https://mdoege.github.io/PySynth/ [Data de consulta: Maig 2018]

[27] A. Tanbakuchi, "Optical music recognition: state-of-the-art and open issues," Febrer 2016. [En línia]. Disponible: http://tanbakuchi.com/posts/comparison-of-openv-interpolation-algorithms/

[28] Opencv - detecció de patrons. [En línia]. Disponible: https://docs.opencv.org/2.4/modules/imgproc/doc/object_detection.html [Data de consulta: Març 2018]

[29] Especificacions dels valors de l'estàndard midi. [En línia]. Disponible: https://en.wikipedia.org/wiki/MIDI_tuning_standard [Data de consulta: Abril 2018]

[30] Software sintetitzador musical. [En línia]. Disponible: https://sourceforge.net/projects/timidity/ [Data de consulta: Abril 2018]

[31] Framework web2py. [En línia]. Disponible: http://www.web2py.com/ [Data de consulta: Abril 2018]

[32] Framework django. [En línia]. Disponible: https://www.djangoproject.com/ [Data de consulta: Abril 2018]

[33] Framework wordpress. [En línia]. Disponible: https://es.wordpress.com/ [Data de consulta: Abril 2018]

[34] J. Colao, "With 60 million websites, wordpress rules the web. so where's the money?" *Forbes*, Setembre 2012. [En línia]. Disponible: https://www.forbes.com/sites/jjcolao/2012/09/05/the-internets-mother-tongue/#5e48546169f6

[35] Llicència wordpress. [En línia]. Disponible: https://wordpress.org/about/license/ [Data de consulta: Abril 2018]

[36] Entorn de desenvolupament en línia i servei de desenvolupament web, pythonanywhere. [En línia]. Disponible: https://www.pythonanywhere.com/ [Data de consulta: Abril 2018]

[37] Web creada per utilitzar l'aplicació music scanner. [En línia]. Disponible: http://joanbruch.pythonanywhere.com/ [Data de consulta: Maig 2018]

[38] Explicació sobre com utilitzar protocol https a pythonanywhere. [En línia]. Disponible: https://help.pythonanywhere.com/pages/ForcingHTTPS [Data de consulta: Maig 2018]

[39] Entrada de la wikipèdia: Protocol https. [En línia]. Disponible: https://en.wikipedia.org/wiki/HTTPS [Data de consulta: Maig 2018]

[40] Eina per escanejar la seguretat d'una pàgina web. [En línia]. Disponible: https://www.ssllabs.com/ssltest/ [Data de consulta: Maig 2018]

[41] Recaptcha. eina per controlar els bots a les webs. [En línia]. Disponible: https://www.google.com/recaptcha/intro/v3beta.html [Data de consulta: Maig 2018]

[42] Document sobre la usabilitat. [En línia]. Disponible: http://ioc.xtec.cat/materials/FP/Materials/2252_DAM/DAM_2252_M07/web/html/ [Data de consulta: Maig 2018]

APÈNDIX A

La Música: definicions i conceptes importants

En aquest apartat explicaré els conceptes musicals més importants per tal d'entendre millor tot el relacionat amb la música que apareix en aquest treball.

Definició 1. La música és un art que s'expressa mitjançant la combinació de sons, tenint com a elements constitutius la melodia, l'harmonia, el ritme i el timbre.

A.1. La melodia i l'harmonia

La melodia indica com sona una peça musical. Recull el seu tema i és el que les persones acostumem a recordar d'una peça.
Més específicament, la melodia inclou patrons de canvi entre tons i duracions; i de manera més general, inclou qualsevol patró d'interacció entre notes i la seva durada en el temps.
L'harmonia musical és tot el que es relaciona amb els sons simultanis. Es pot entendre com l'aspecte "vertical"de la música, mentre que la melodia en seria l'aspecte "horitzontal". Molt sovint l'harmonia és el resultat de tocar diverses línies melòdiques o motius alhora.
L'aplicació que he desenvolupat té la capacitat de reconèixer partitures d'una sola línia melòdica, per tant, les peces amb més d'una nota (A.1.1) existents al mateix temps no les podrà identificar correctament.

Aquesta imatge és un fragment d'una partitura de la cançó de Pirates del Carib, per piano. Es pot apreciar molt bé la diferència entre la melodia (línia superior) i l'harmonia (línia inferior).

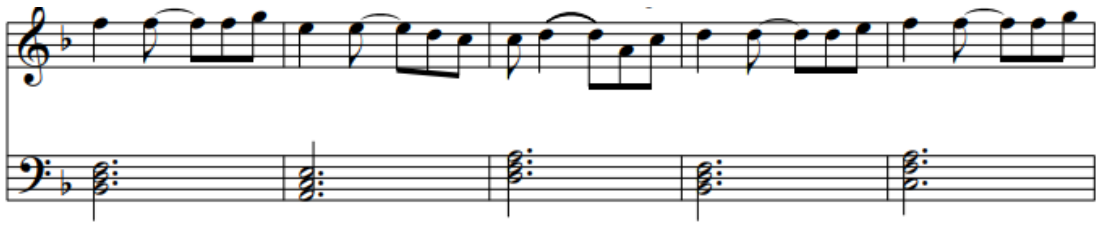


Figura A.1: Fragment partitura de Pirates del Carib.
Font: https://trinthepianist.com/

A.1.1. Les notes

Una nota és un so determinat per una vibració que té una freqüència fonamental constant. Així doncs, el terme *nota musical* fa referència a un so amb una determinada freqüència en si; mentre que per a definir el signe que s'utilitza en la notació musical per representar l'altura i la durada relativa d'un so, s'anomena *figura musical*.

Les notes s'escriuen al pentagrama (a les línies, espais o amb línies extres superiors o inferiors).

Després de diverses reformes i modificacions, les notes van passar a ser les que es coneixen actualment:

- do, re, mi, fa, sol, la, si (segons el sistema de notació llatí).
- C, D, E, F, G, A, B (segons el sistema de notació anglès).

«La 440» és el nom que se li dóna al so que produeix una vibració a 440 Hz a 20°C i serveix com a estàndard de referència per afinar l'altura musical.

El «La 440» és la nota la o A que trobem situada al segon espai del pentagrama (A.1.2) de clau de sol (A.1.3).

Amb la següent fórmula es calculen les freqüències fonamentals de la resta de notes:

$$f = 2^{n/12} x 440 Hz \tag{A.1}$$

A.1.2. Pentagrama

El pentagrama és el lloc on s'escriuen les notes i tots els altres signes musicals en el sistema de notació musical occidental. Està format per cinc línies horitzontals i quatre espais equidistants que s'enumeren de baix a dalt.

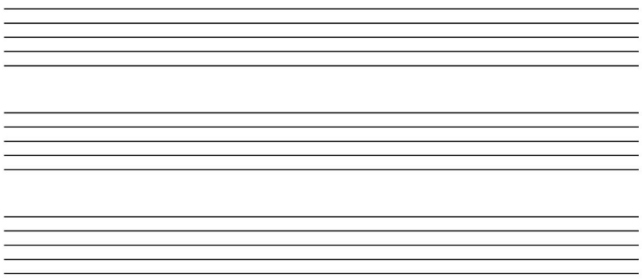


Figura A.2: Tres línies de pentagrama. **Font:** Pròpia

A.1.3. Claus

En notació musical, la clau és un signe que té la funció d'indicar l'altura de les notes escrites. Si està situada sobre una de les línies al principi del pentagrama, indica el nom i l'altura de les notes d'aquella línia, que es pren com a punt de referència per establir els noms i altures de la resta de les notes d'aquell pentagrama. Tot i que no és habitual, la clau pot canviar-se en qualsevol punt de l'obra si es requereix.
Les claus més utilitzades, i per ordre de la imatge són:

1. Clau de sol (en segona)
2. Clau de do en tercera
3. Clau de do en quarta
4. Clau de fa (en quarta)

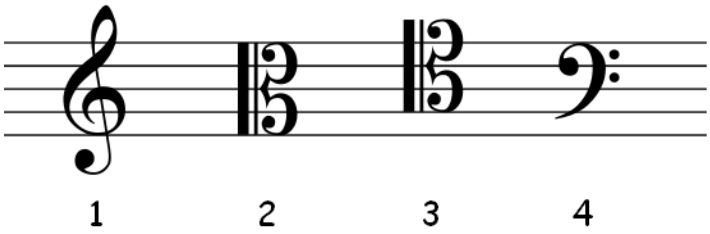


Figura A.3: Les claus més utilitzades. **Font:** Vikipèdia

A.1.4. Alteracions i armadura

Les alteracions són uns símbols gràfics que col · loquem al costat esquerre de les notes musicals. Aquestes alteracions modifiquen l'altura de la nota.

Símbol	Nom de l'alteració	Descripció
♯	Sostingut	Eleva la nota musical un semito
♭	Bemoll	Disminueix la nota musical un semito
♮	Becaire	Anul·la l'efecte de qualsevol alteració anterior
𝄪	Doble sostingut	Eleva la nota un to
𝄫	Doble bemoll	Disminueix la nota musical un to

Taula A.1: Les claus més utilitzades. **Font:** Vikipèdia

Definició 2. Un **semitò** és la distància mínima de separació entre dues notes diferents. Es troba entre les notes mi-fa i si-do.

Definició 3. Un **to** és la distància doble del semitò. Es troba entre el do-re, re-mi, fa-sol, la-si.

Podem classificar les alteracions en accidentals o fixes, en funció de si apareixen de manera arbitrària per la partitura o si apareixen al principi de la línia de pentagrama, entre la clau (A.1.3) i el compàs (A.2.2).

Les alteracions que apareixen a l'armadura (siguin sostinguts o bemolls) sempre apareixen amb un ordre definit.
L'ordre complet dels sostinguts és: fa, do, sol, re, la, mi, si. I l'ordre dels bemolls és: si, mi, la, re, sol, do, fa.

A.2. El ritme

El ritme és l'organització de la música en el temps i com a successió de sons i silencis en funció de les respectives durades, que acostumen a regir-se per un sistema de proporcionalitat que pren com a patró de mesura la pulsació.
Vinculats al ritme hi ha el *tempo*, que acaba determinant la durada absoluta de cada figura, i la *mètrica*, íntimament lligada al ritme tot i que no té a veure amb les durades sinó amb els accents i per tant, amb la intensitat. Existeixen ritmes diferents, en funció de la seva durada i de si són silenci o so.

A.2.1. El Tempo i la pulsació

El tempo és la velocitat a la qual s'interpreta una composició musical. És una paraula italiana que literalment significa 'temps'. Es mesura en pulsacions per minut (ppm). En funció del tempo, una mateixa obra musical té una durada més o menys llarga; de la mateixa manera que cada figura musical no té una durada especifica i fixa en segons, sinó que en funció del tempo durarà més o menys temps.
El nom de pulsació suggereix la idea de batec regular. És la unitat de temps segons la qual s'estructura un discurs musical, independentment de ritmes, compassos i accents. A vegades, quan escoltem música, de manera inconscient marquem la pulsació amb el peu o la mà. Les línies de color que hi ha sota les notes de la imatge representen la pulsació.

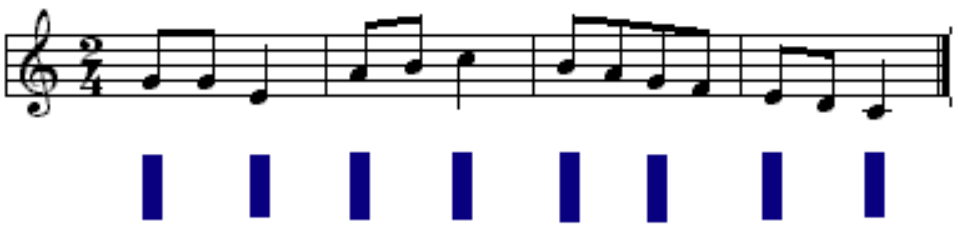
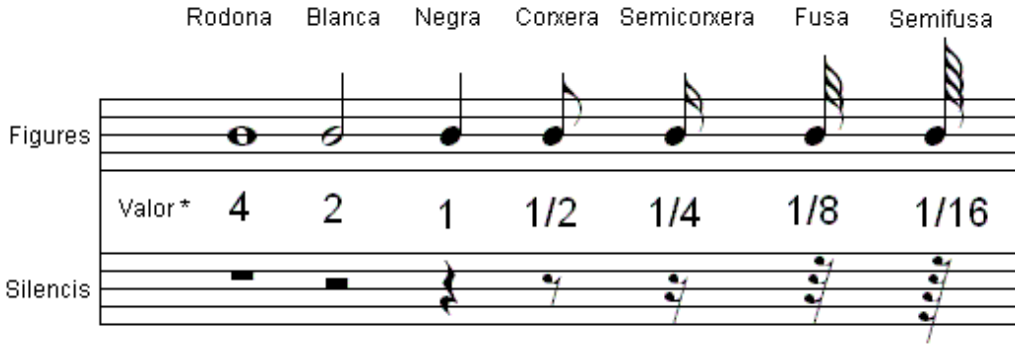


Figura A.4: Melodia amb la pulsació marcada.
Font: http://grups.blanquerna.url.edu/m11/infantil/continguts.htm

A la següent taula hi ha figures i silencis ordenats de més a menys durada, tenint en compte la pulsació com a unitat de referència.



Taula A.2: Taula de figures i silencis amb el respectiu valor.
Font: http://grups.blanquerna.url.edu/m11/infantil/continguts.htm

D'aquesta taula observem que la rodona és l'element inicial amb el qual es van subdividint la resta de figures i silencis. Així, una blanca val la meitat que una rodona, una negra val la meitat que una blanca i una quarta part que la rodona,... Tot i que quan afegim un punt (·) al costat de cada figura, el valor s'incrementa la meitat del que li correspon. Per exemple, una blanca amb un punt valdrà 3 (2+1) enlloc de 2, i una corxera valdrà 3/4 (1/2 + 1/4) enlloc de 1/2.
Les figures rítmiques de valor corxera o inferior és habitual trobar-les escrites de forma ajuntada:

SÍMBOL	NOM TEÒRIC
	Negra
	Dues corxeres
	Silenci de negra
	Quatre semicorxeres
	Blanca
	Corxera, dues semicorxeres
	Dues semicorxeres, corxera
	Negra en punt, corxera
	Corxera, negra, corxera
	Corxera en punt, semicorxera

Taula A.3: Símbols rítmics més habituals.
Font: http://grups.blanquerna.url.edu/m11/infantil/continguts.htm

L'aplicació desenvolupada no és capaç de reconèixer les figures rítmiques que tenen punts (els punts els ignora). També reconeix malament les corxeres escrites de forma individual (les converteix a negres).

A.2.2. La mètrica i el compàs

La mètrica és l'estructura subjacent en la música que es basa en l'aparició periòdica, normalment a intervals regulars, de sons o altres elements accentuats.
El compàs és l'entitat mètrica musical, composta per diverses unitats de temps. Aquesta divisió es representa gràficament des del segle XVI per unes línies verticals, anomenades línies divisòries o barres de compàs que es col · loquen perpendicularment a les línies del pentagrama.

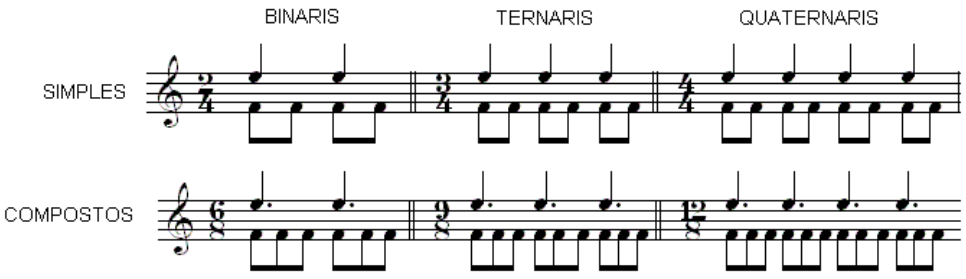
Segons la quantitat de parts o pulsacions de què consten els compassos, es clas-

sifiquen en compassos binaris, si tenen dues pulsacions, ternaris, si en tenen tres, o quaternaris, si en tenen quatre.
També existeixen altres tipus de compassos amb nombre superior de pulsacions que sovint s'entenen com derivats dels tres tipus bàsics (binari, ternari i quaternari).

Representació dels compassos

El símbol que expressa el compàs sempre adopta forma de fracció. En el cas dels compassos simples el numerador expressa el nombre de pulsacions que té cada compàs, mentre que en el cas dels compassos compostos expressa el total de parts ternàries de pulsació que conté el compàs. El nombre de pulsacions d'un compàs compost és el resultat de dividir el numerador per tres.

El denominador, en el cas dels compassos simples expressa la figura que serveix com a unitat de pulsació: 2 = blanca; 4 = negra, 8 = corxera, són les més habituals. En el cas dels compassos compostos el denominador expressa la figura que serveix com a unitat de part ternària de pulsació, de manera que la unitat de pulsació és la que equivaldria a tres d'aquestes figures.
En aquesta taula es mostren els tres tipus de compassos bàsics (simples) amb la negra com a unitat de pulsació, i els seus respectius compassos compostos:



Taula A.4: Compassos més habituals.
Font: http://grups.blanquerna.url.edu/m11/infantil/continguts.htm

L'aplicació desenvolupada no reconeix compassos; a l'hora de crear la partitura, la genera sobre una partitura de compàs simple quaternari.

APÈNDIX B

Utilitat dels patrons

He creat diferents patrons d'elements musicals per utilitzar-los comparant la partitura a escanejar amb aquests patrons.
Per crear-los he utilitzat el programa MuseScore [20] i imatges d'Internet.

B.1. Com crear un nou patró

En aquest apartat, explicaré com incloure un nou patró d'un element musical al programa.
Agafem d'exemple l'element musical bemoll.
Amb el programa MuseScore, creem una nova partitura, afegim l'element bemoll a la partitura i fem una captura d'imatge del bemoll procurant que sigui com més retallada i ajustada possible (si és necessari, amb un editor d'imatges l'acabem de retallar). Des d'Internet, també podem buscar imatges de bemolls i descarregar les que creguem convenients en funció de la resolució, mida i visualització.
Un cop hem guardat els fitxers patró a la carpeta de patrons corresponent, ja podem començar a escriure les següents línies de codi:

```
#Definim una llista de patrons de cada element
bemoll_fitxers = [
"patrons/bemoll-linia.png",
"patrons/bemoll-espai.png" ]
#Carreguem les imatges patró
bemoll_imgs = []
for bemoll_fitxer in bemoll_fitxers:
        bemoll_imgs.append(cv2.imread(bemoll_fitxer, 0))
#Definim constants per realitzar l'escalat
```

```
bemoll_inferior, bemoll_superior, bemoll_llindar = 50, 180, 0.77


#Comprovem les coincidèndies dels bemolls amb la partitura
rects_bemoll = ubica_patro(img_partitura_bw, bemoll_imgs,
                           bemoll_inferior, bemoll_superior, bemoll_llindar)


#Fusionem els resultats dels bemolls
llindar_fusio = 0.5
rects_bemoll =
        fusiona_rects([j for i in rects_bemoll for j in i], llindar_fusio)


#Dibuixem els rectangles coincidents a la partitura original
rects_bemoll_img = img_partitura.copy()
for r in rects_bemoll:
    r.dibuixa(rects_bemoll_img, (0, 0, 255), 2)
cv2.imwrite('rects_bemoll_img.png', rects_bemoll_img)
```

El resultat després d'executar el programa és:

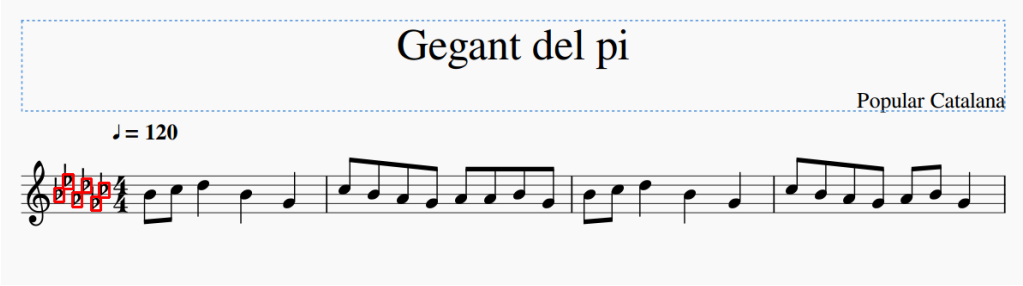


Figura B.1: Bemolls escanejats **Font:** Pròpia.

B.2. Patrons Utilitzats

A l'apartat anterior he explicat com crear i utilitzar un patró pel reconeixement. En aquest apartat mostro tots els patrons que he utilitzat en el programa:

B.2.1. Pentagrama

El patró del pentagrama cal que sigui com més estret millor perquè si fós ample, no es detectarien els pentagrames entre dues notes separades però a una curta distància entre elles.

Figura B.2: Patró del pentagrama

B.2.2. Negres

Hi ha dos patrons de negres diferents. Un patró té el fons blanc i l'altre no té fons d'imatge.

(a) Negra amb fons blanc

(b) Negra sense fons

Figura B.3: Patrons de negres

B.2.3. Blanques

Hi ha quatre patrons de blanca diferents. Blanques a una línia de pentagrama o a un espai de pentagrama i amb el fons blanc o sense fons d'imatge.

(a) Situada als espais

(b) Situada a les línies

Figura B.4: Patrons de blanques amb fons blanc

(a) Situada als espais

(b) Situada a les línies

Figura B.5: Patrons de blanques sense fons

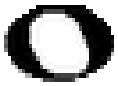
B.2.4. Rodones

Hi ha quatre patrons de rodona diferents. Rodones a una línia de pentagrama o a un espai de pentagrama i amb el fons blanc o sense fons d'imatge.

(a) Situada als espais

(b) Situada a les línies

Figura B.6: Patrons de rodones amb fons blanc



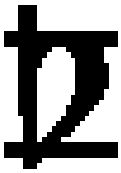
(a) Situada als espais

(b) Situada a les línies

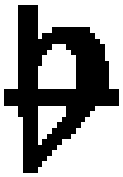
Figura B.7: Patrons de rodones sense fons

B.2.5. Bemolls

Hi ha dos patrons de bemolls diferents. Bemoll entre una línia i bemoll entre un espai del pentagrama.



(a) Espai



(b) Línia

Figura B.8: Patrons de bemolls

B.2.6. Sostinguts

Hi ha quatre patrons de sostinguts diferents. Sostinguts a una línia de pentagrama o a un espai de pentagrama i amb el fons blanc o sense fons d'imatge.

(a) Situat als espais **(b)** Situat a les línies

Figura B.9: Patrons de sostinguts sense fons

(a) Situat als espais **(b)** Situat a les línies

Figura B.10: Patrons de sostinguts amb fons blanc

B.2.7. Silenci de negra

El silenci de negra l'he inclòs en els patrons de reconeixement. A les partitures de test es pot comprovar que es reconeixen però que no es transformen a MIDI perquè no he trobat la opció de fer-ho amb la llibreria MIDIUtil.

(a) **(b)**

Figura B.11: Patrons de silenci de negra

APÈNDIX C

Programa MuseScore

A continuació explicaré els passos a realitzar per escriure una partitura amb el programa MuseScore:

Obrim el programa MuseScore i omplim els camps que ens apareixen en pantalla, seguint les instruccions de l'assistent.

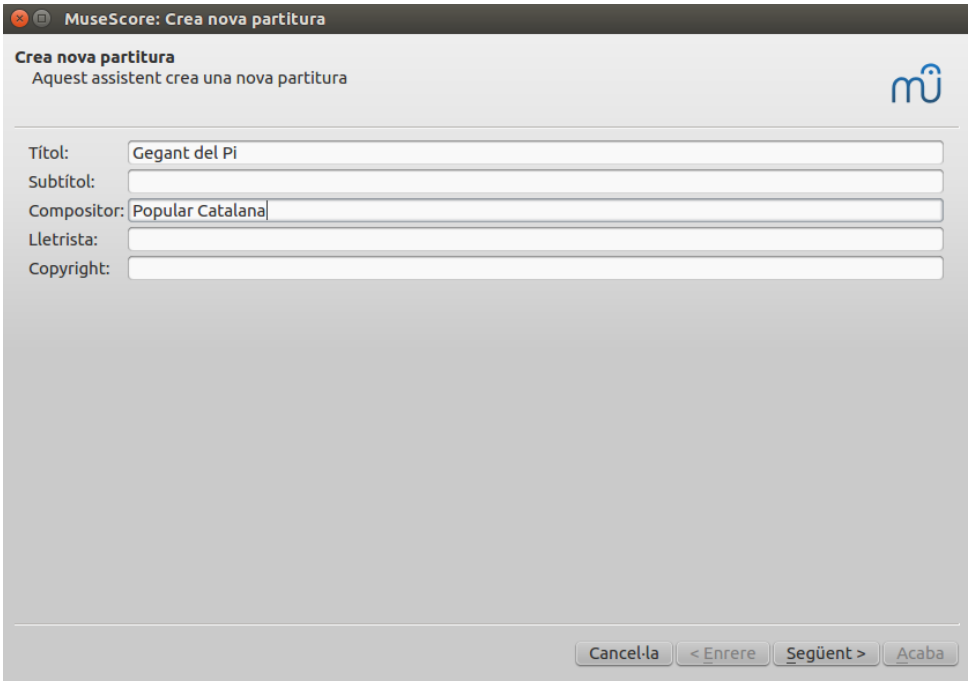


Figura C.1: Obrir document nou **Font:** Elaboració pròpia.

El següent pas és seleccionar la plantilla de partitura necessària pel que volem escriure:

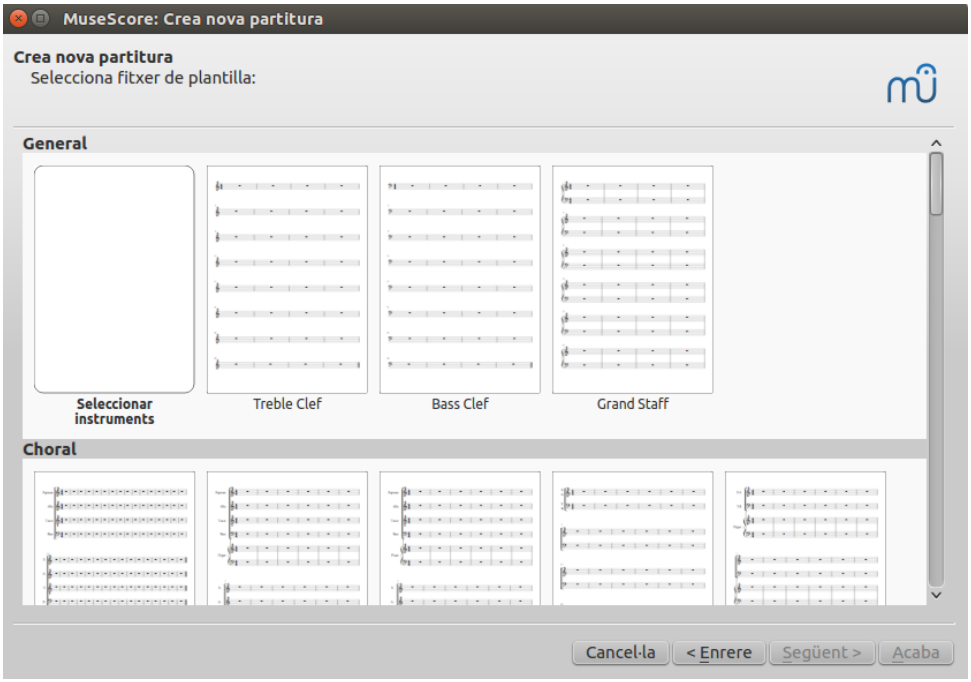


Figura C.2: Plantilles disponibles **Font:** Elaboració pròpia.

A continuació seleccionem el tempo i l'armadura:

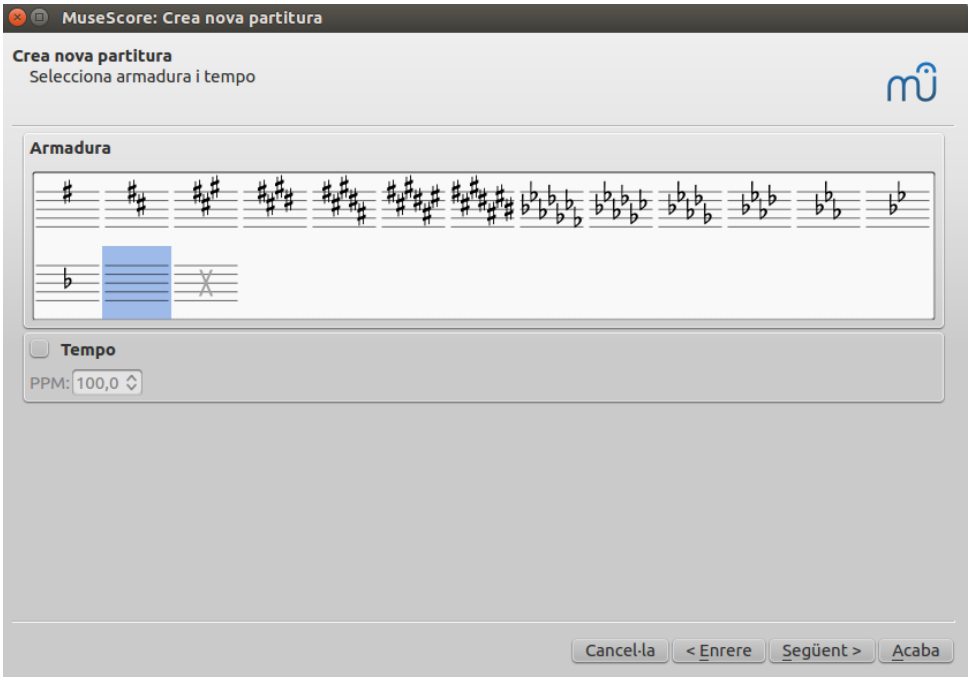


Figura C.3: Escollir armadura i tempo **Font:** Elaboració pròpia.

Finalment escollim els paràmetres relacionats amb el compàs: Indicació de compàs, si hi ha anacrusa i el nombre total de compassos.

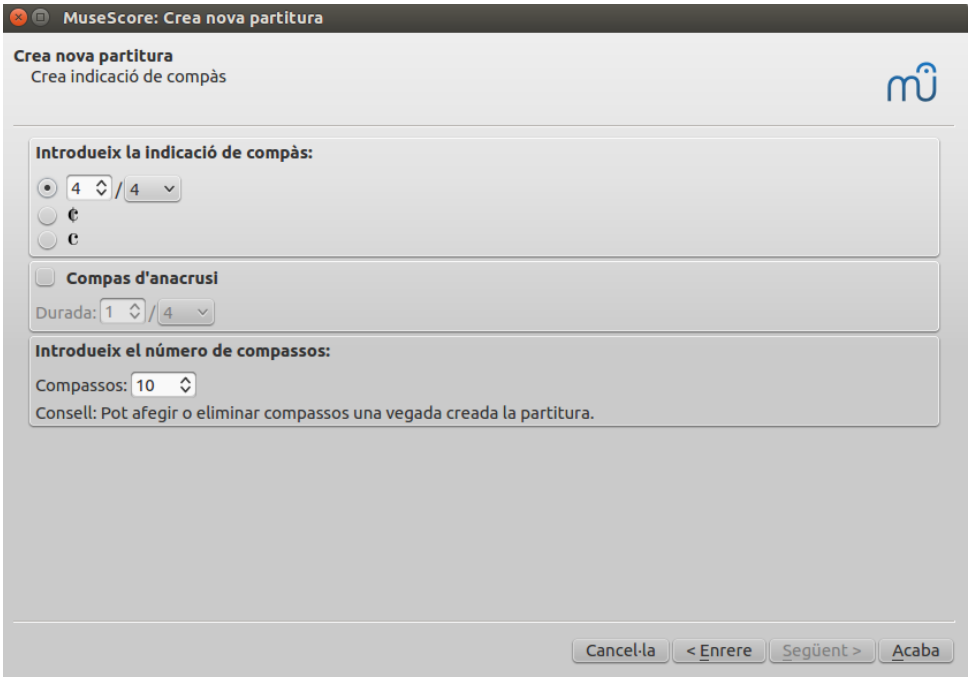


Figura C.4: Escollir el compàs **Font:** Elaboració pròpia.

Finalment, ja podem començar a escriure els elements musicals a la plantilla de partitura generada amb l'assistent del MuseScore.

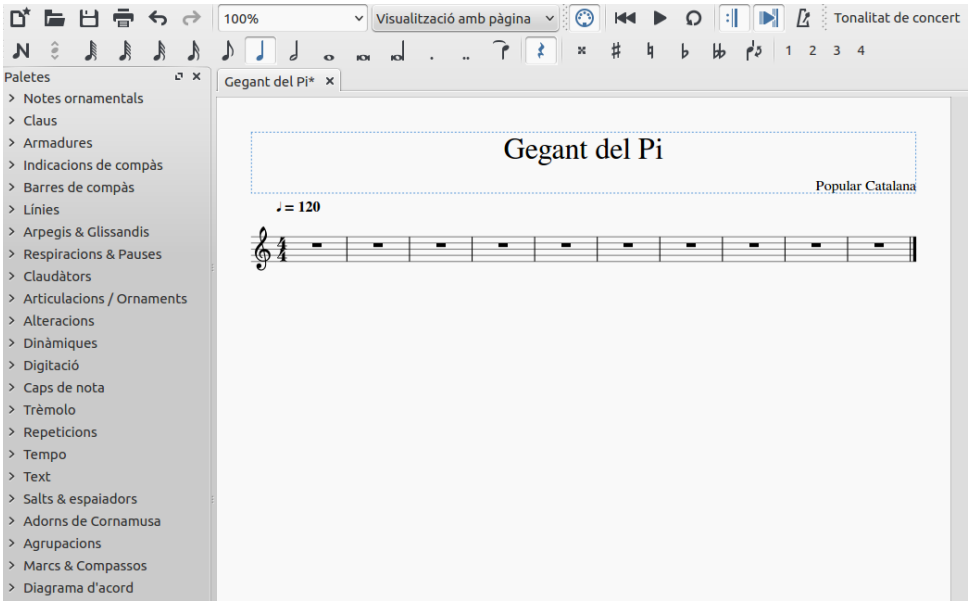


Figura C.5: Fi de l'assistent **Font:** Elaboració pròpia.

És important saber que amb les *Paletes* que apareixen a la banda esquerra de la pantalla del programa, podem modificar tots els paràmetres introduïts amb

l'assistent al crear una partitura nova.

El següent pas és seleccionar un valor de ritme concret, clicar la icona superior esquerra ,(damunt de *Paletes*), que activa el mode d'entrada de notes a la partitura i et permet començar a escriure.

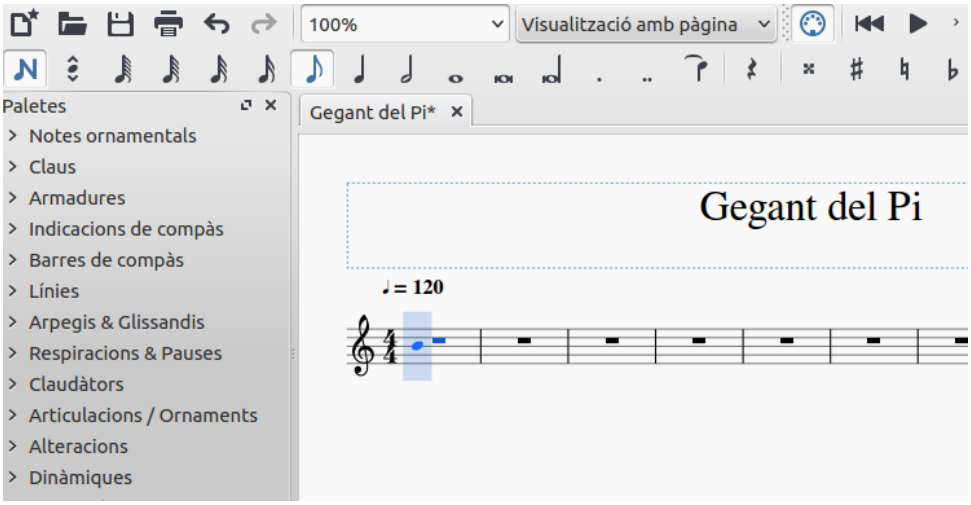


Figura C.6: Escriure una nota **Font:** Elaboració pròpia.

Ens col · loquem sobre el pentagrama i cliquem damunt l'alçada de la nota concreta. Ja hem escrit el primer compàs:

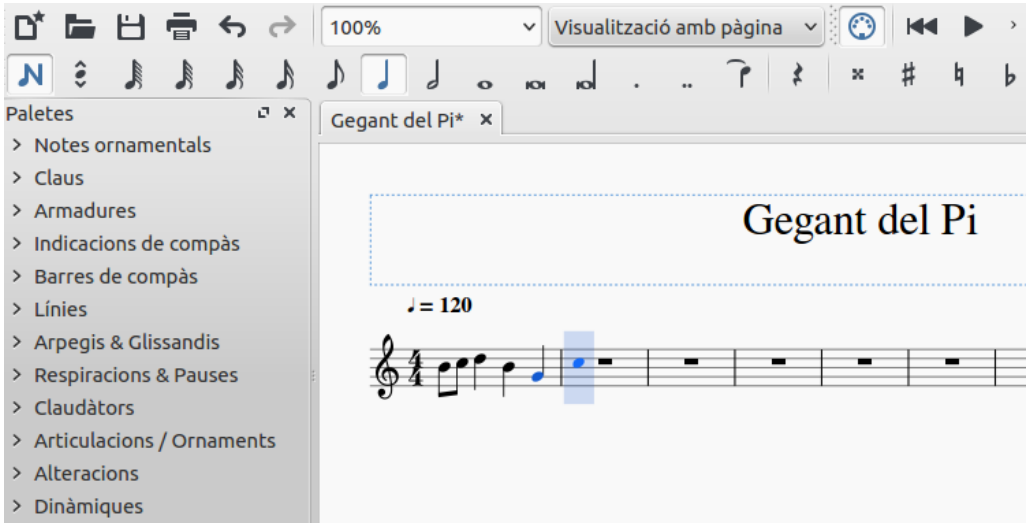


Figura C.7: Primer compàs escrit **Font:** Elaboració pròpia.

Seguim entrant notes fins acabar d'escriure la partitura sencera.

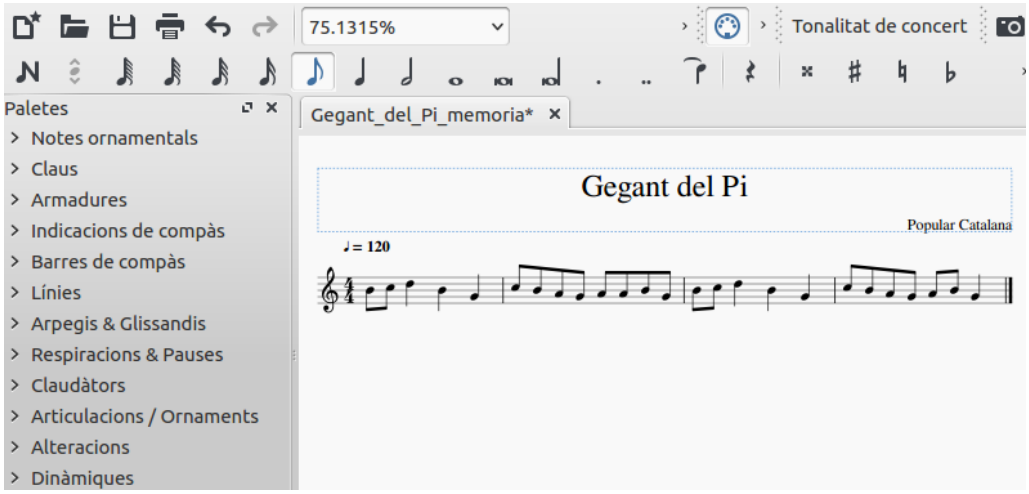


Figura C.8: Partitura completa **Font:** Elaboració pròpia.

APÈNDIX D

Proves realitzades

A continuació mostro les imatges originals, les imatges amb els patrons reconeguts dibuixats i la captura de la partitura transformada a format MIDI dels tests que he realitzat.

Una mateixa cançó pot estar escrita diferent encara que sonin les mateixes freqüències de notes (utilitzant alteracions). Per això, hi ha algunes captures del resultat MIDI que visualment són molt diferents de l'original però auditivament (malgrat no es pot representar amb un treball escrit), són molt semblants.

D.1. Partitures escrites personalment amb el programa MuseScore

Gegant del Pi en Do Major

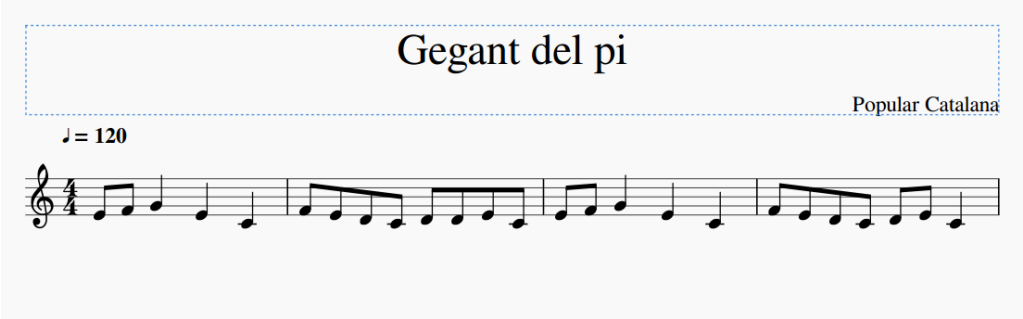


Figura D.1: Partitura Original **Font:** Pròpia

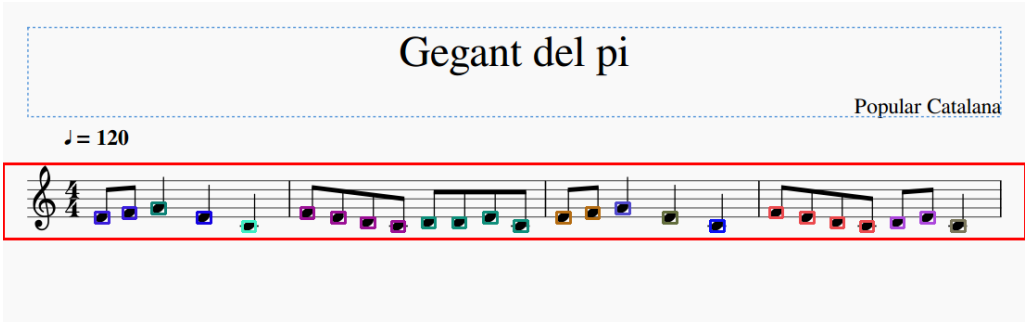


Figura D.2: Partitura amb els patrons reconeguts dibuixats **Font:** Pròpia

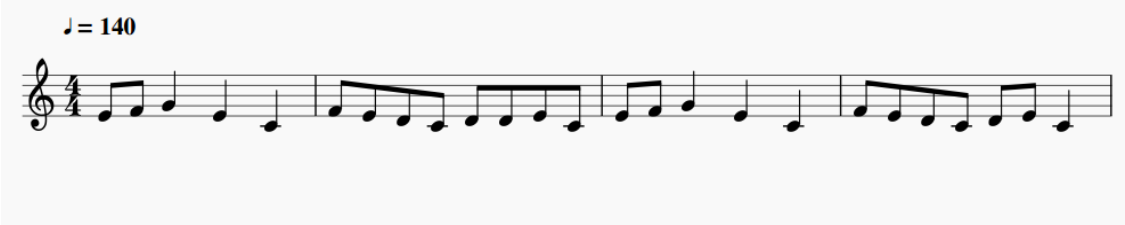


Figura D.3: Captura de pantalla de la partitura MIDI **Font:** Pròpia

Comentari: Partitura reconeguda al 100 %.

Gegant del Pi en Re Major

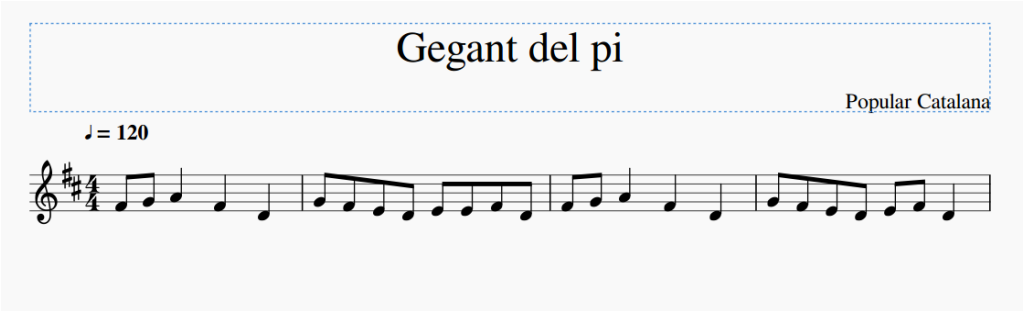


Figura D.4: Partitura Original **Font:** Pròpia

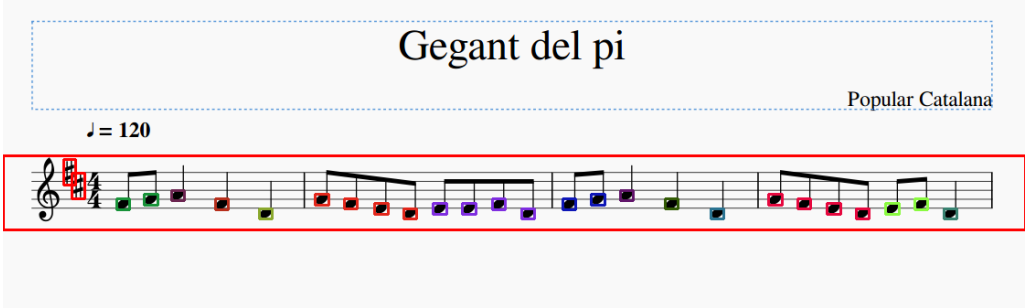


Figura D.5: Partitura amb els patrons reconeguts dibuixats **Font:** Pròpia

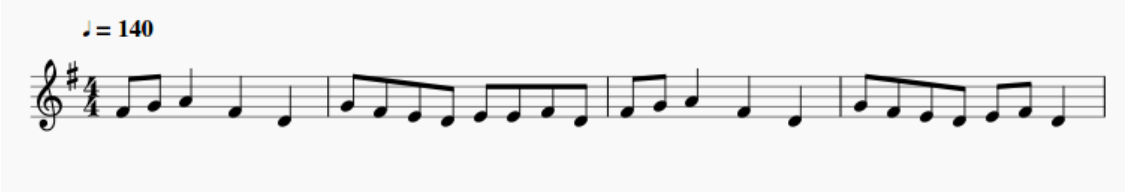


Figura D.6: Captura de pantalla de la partitura MIDI **Font:** Pròpia

Comentari: Partitura reconeguda al 100 %.

Gegant del Pi en Mi Major

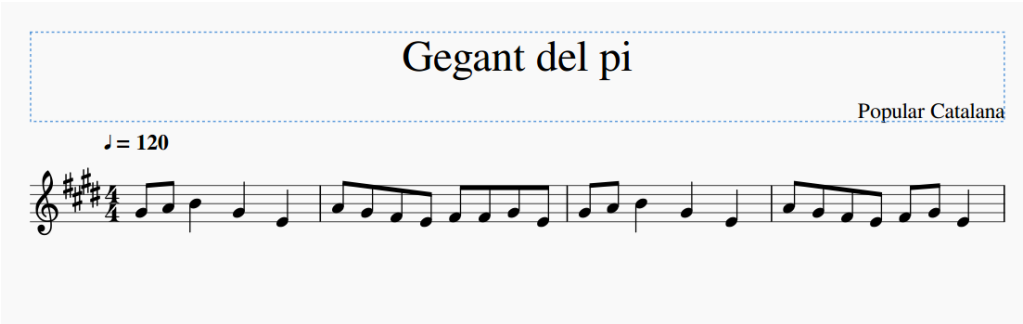


Figura D.7: Partitura Original **Font:** Pròpia

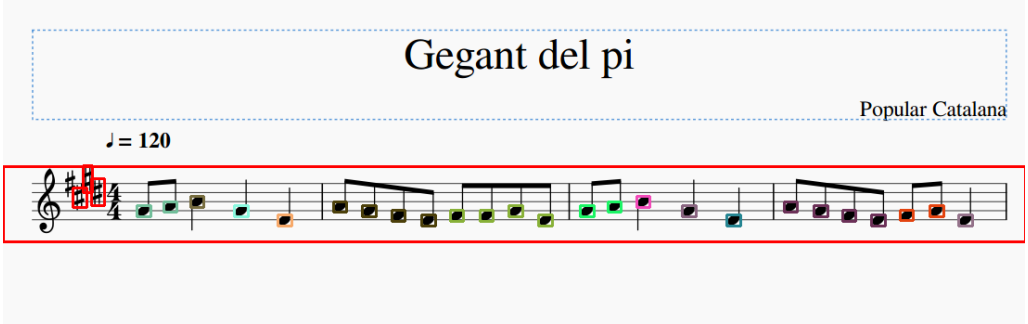


Figura D.8: Partitura amb els patrons reconeguts dibuixats **Font:** Pròpia

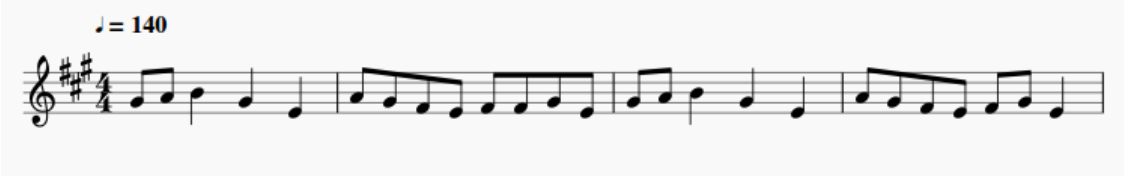


Figura D.9: Captura de pantalla de la partitura MIDI **Font:** Pròpia

Comentari: Partitura reconeguda al 100 %. Per com està dissenyat l'algorisme de reconeixement de les alteracions, podem veure que ha reconegut les últimes tres alteracions (do,sol,re) però a l'algorisme no li importa quines alteracions reconeix sinó la quantitat d'alteracions reconegudes. Com que ha reconegut 3 alteracions, totes les notes (fa,do,sol) les ha augmentat un semitò. Coincidències de la partitura, no hi apareix cap nota *Re*, com a conseqüència, l'error en el reconeixement d'alteracions no ha afectat el resultat final.

Gegant del Pi en Fa Major

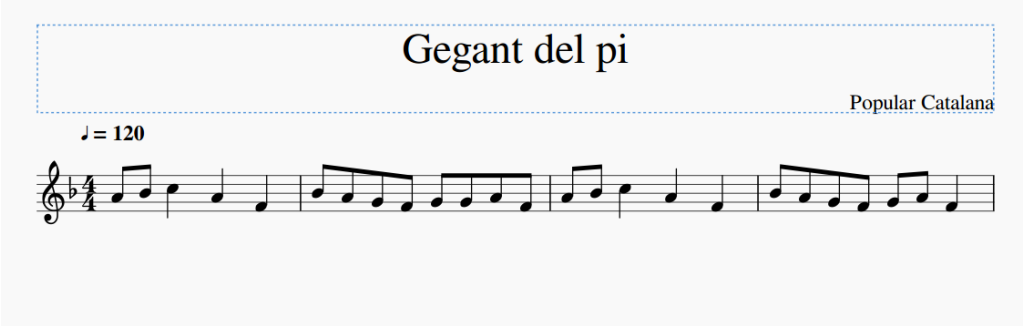


Figura D.10: Partitura Original **Font:** Pròpia

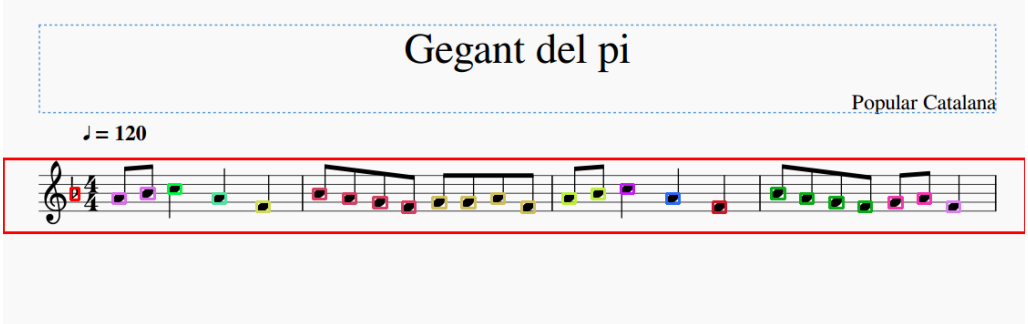


Figura D.11: Partitura amb els patrons reconeguts dibuixats **Font:** Pròpia

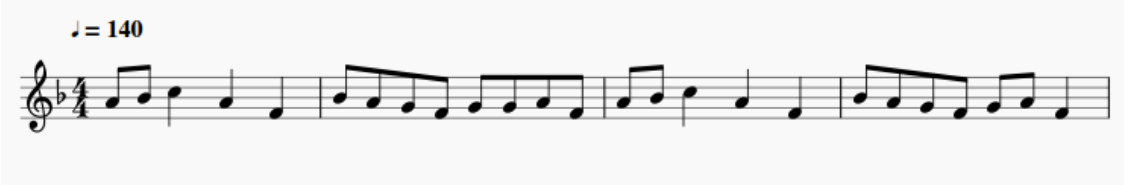


Figura D.12: Captura de pantalla de la partitura MIDI **Font:** Pròpia

Comentari: Partitura reconeguda al 100 %.

Gegant del Pi en Sol Major

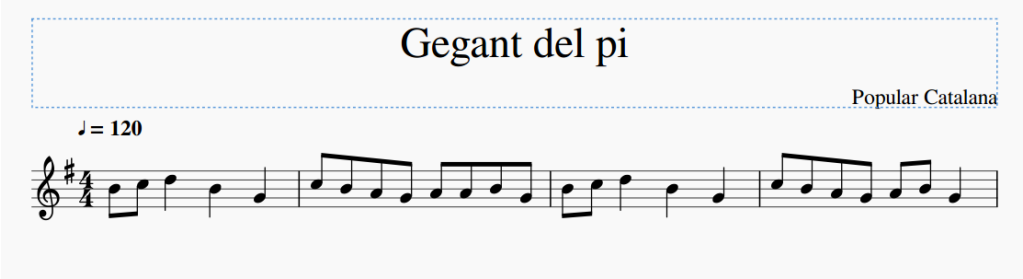


Figura D.13: Partitura Original **Font:** Pròpia

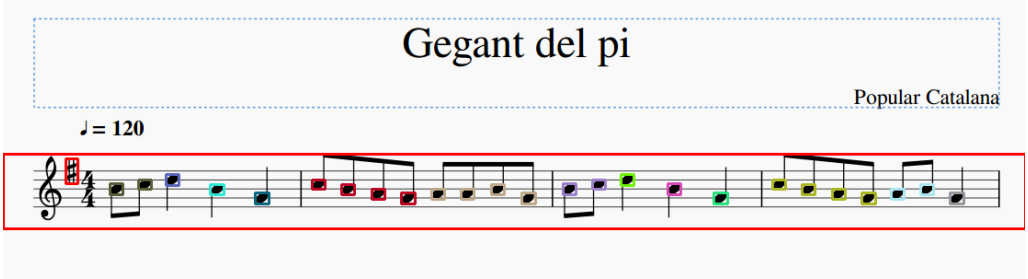


Figura D.14: Partitura amb els patrons reconeguts dibuixats **Font:** Pròpia

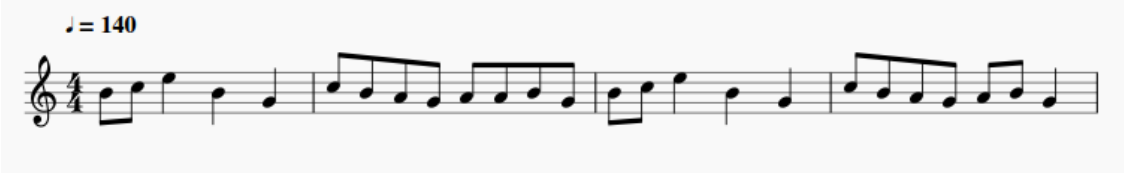


Figura D.15: Captura de pantalla de la partitura MIDI **Font:** Pròpia

Comentari: Partitura reconeguda al 96 %. Ha confós l'altura de les notes Re5 per Mi5.

Gegant del Pi en La Major

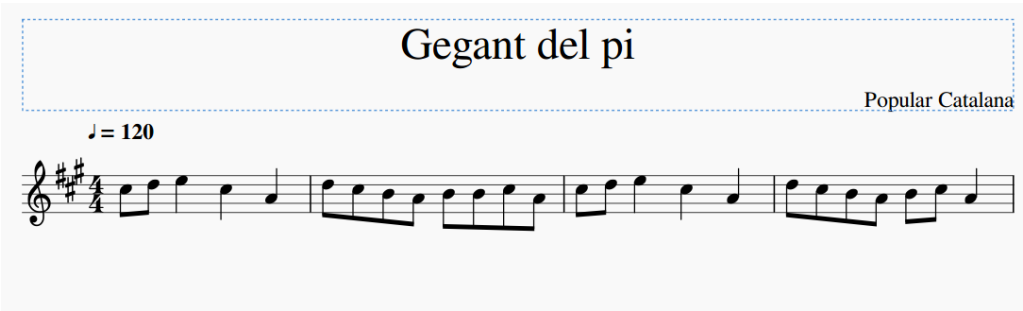


Figura D.16: Partitura Original **Font:** Pròpia

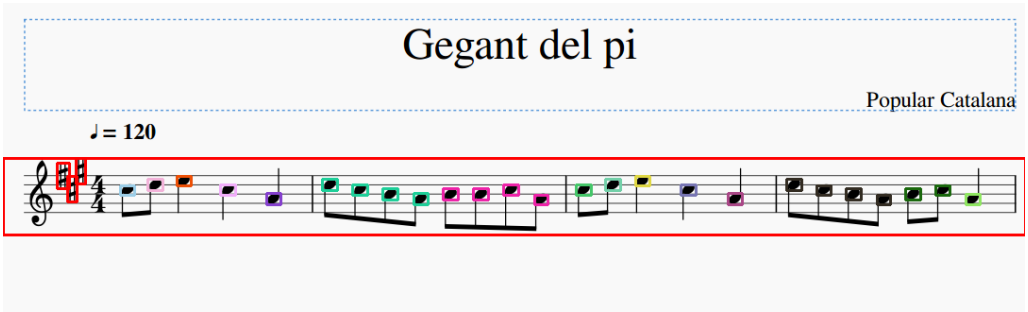


Figura D.17: Partitura amb els patrons reconeguts dibuixats **Font:** Pròpia

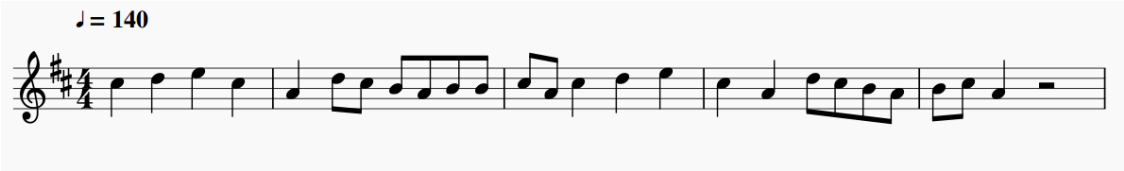


Figura D.18: Captura de pantalla de la partitura MIDI **Font:** Pròpia

Comentari: Partitura reconeguda al 92,86 %. Ha confós les corxeres per negres.

Gegant del Pi en Si Major

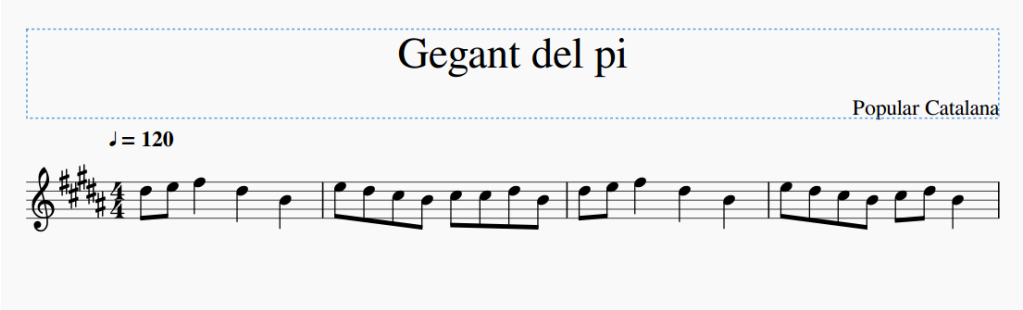


Figura D.19: Partitura Original **Font:** Pròpia

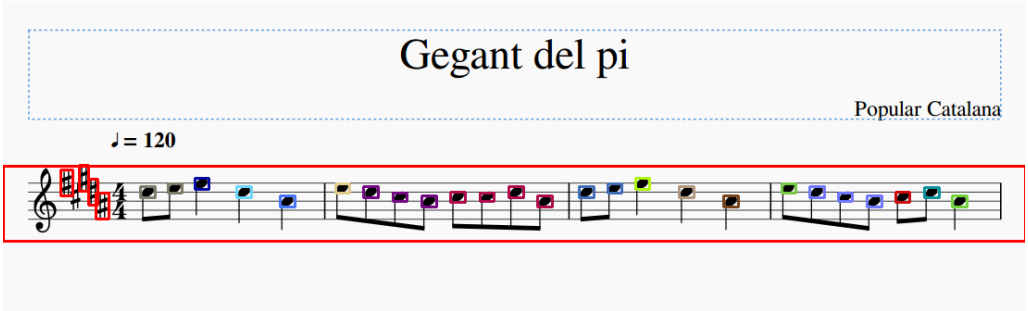


Figura D.20: Partitura amb els patrons reconeguts dibuixats **Font:** Pròpia

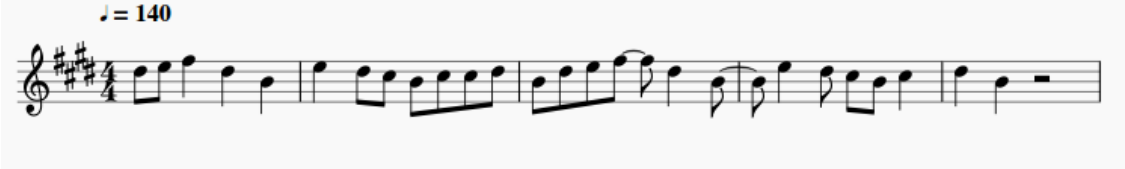


Figura D.21: Captura de pantalla de la partitura MIDI **Font:** Pròpia

Comentari: Partitura reconeguda al 93,33 %. Ha confós les corxeres per negres. En aquesta partitura, igual que per la partitura de Mi Major, ha reconegut un sostingut menys però aquest error no ha afectat al resultat final.

Gegant del Pi en Do bemoll Major

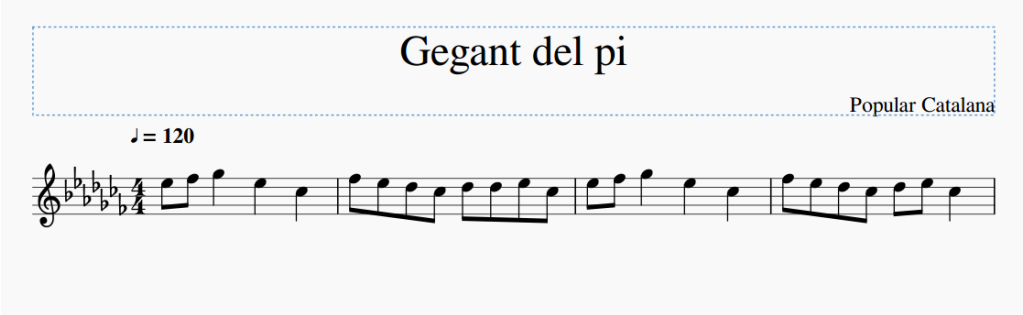


Figura D.22: Partitura Original **Font:** Pròpia

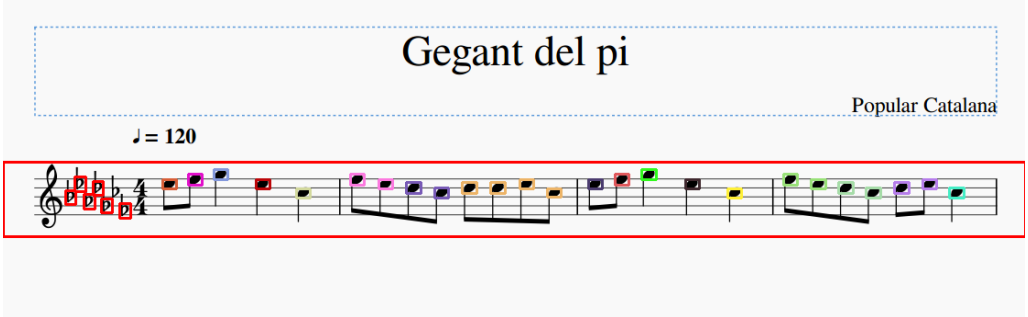


Figura D.23: Partitura amb els patrons reconeguts dibuixats **Font:** Pròpia

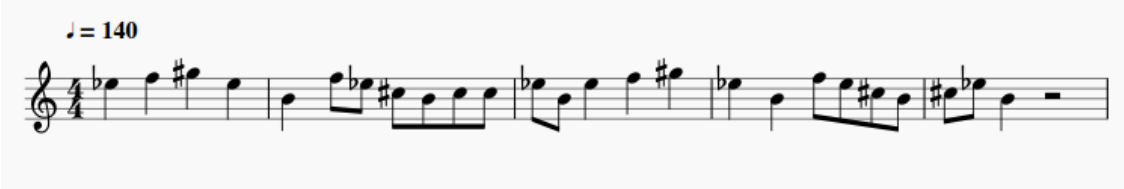


Figura D.24: Captura de pantalla de la partitura MIDI **Font:** Pròpia

Comentari: Partitura reconeguda al 80,19 %. Ha confós les corxeres per negres. En aquesta partitura també ha reconegut un bemoll menys dels que hi ha a l'original i en aquest cas sí que ha afectat el resultat final, ja que totes les notes *Fa* haurien de ser bemolls i són naturals. També ha confós totes les notes Solb per Lab (o Sol#).

Gegant del Pi en Reb Major

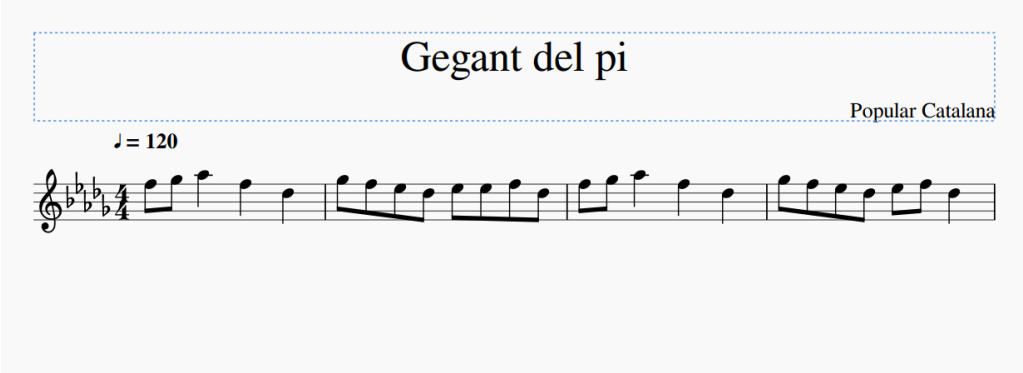


Figura D.25: Partitura Original **Font:** Pròpia

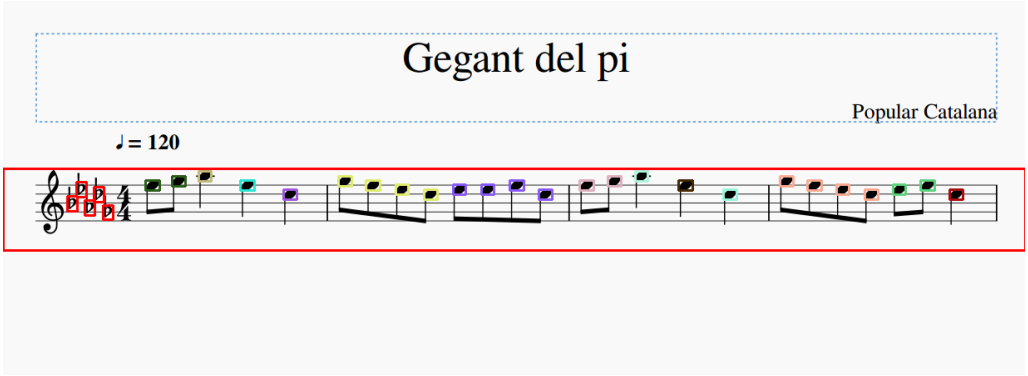


Figura D.26: Partitura amb els patrons reconeguts dibuixats **Font:** Pròpia

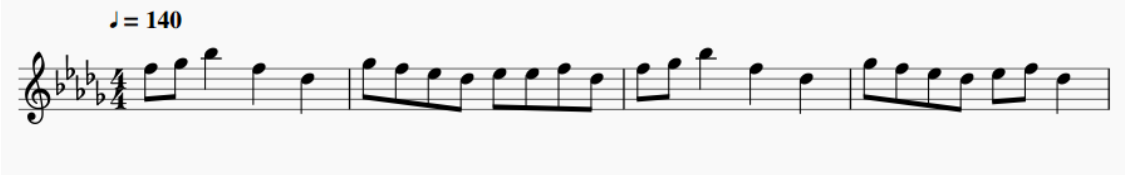


Figura D.27: Captura de pantalla de la partitura MIDI **Font:** Pròpia

Comentari: Partitura reconeguda al 96 %. Ha confós les notes Lab per Sib.

Gegant del Pi en Mib Major - octava 5

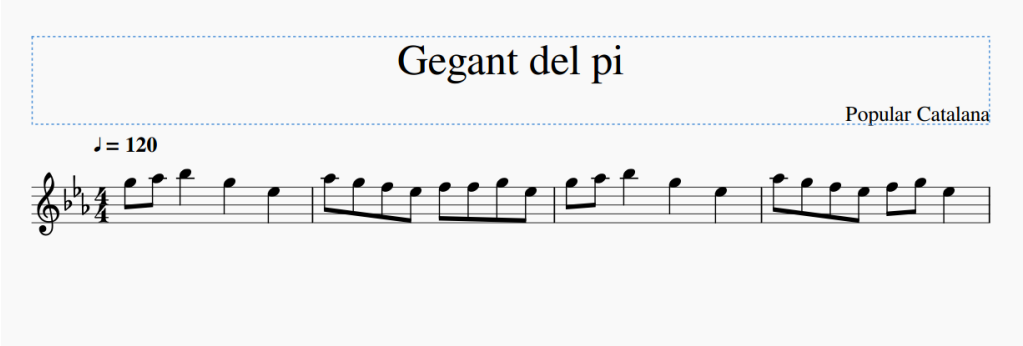


Figura D.28: Partitura Original **Font:** Pròpia

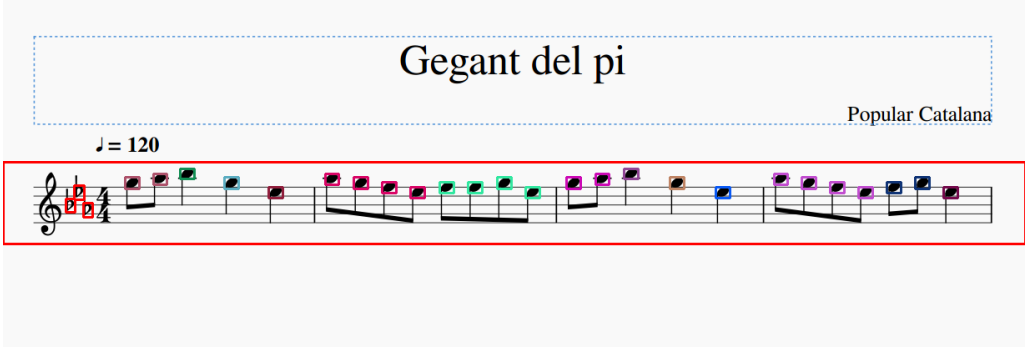


Figura D.29: Partitura amb els patrons reconeguts dibuixats **Font:** Pròpia

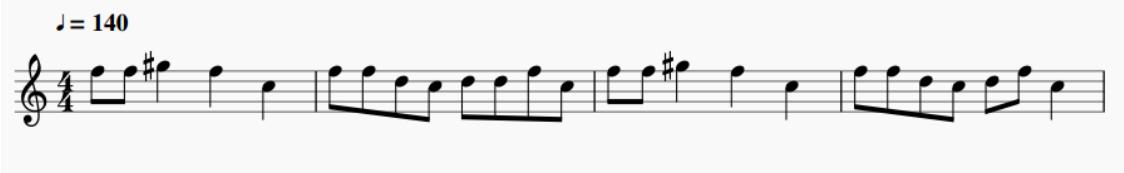


Figura D.30: Captura de pantalla de la partitura MIDI **Font:** Pròpia

Comentari: Partitura reconeguda al 50 %. Al ser notes tan agudes ha reconegut tots els patrons bé però l'altura de les notes no.

Gegant del Pi en Mib Major - octava 4

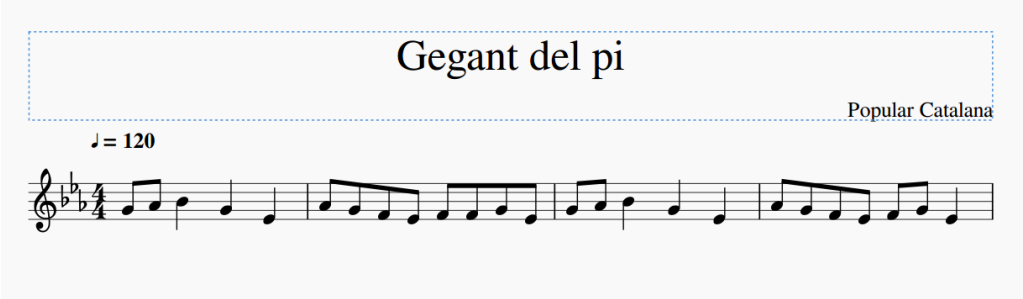


Figura D.31: Partitura Original **Font:** Pròpia

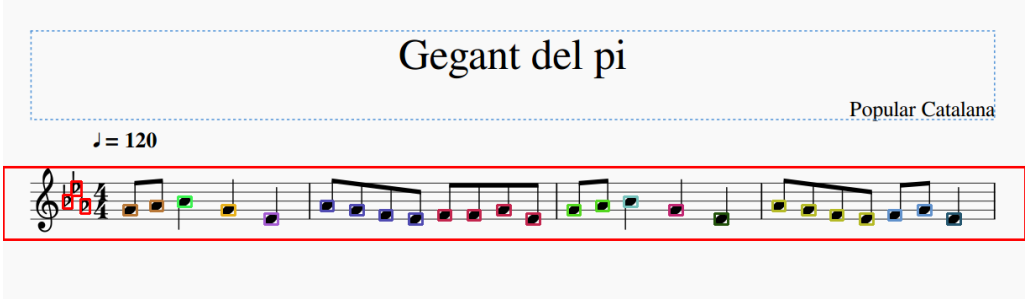


Figura D.32: Partitura amb els patrons reconeguts dibuixats **Font:** Pròpia

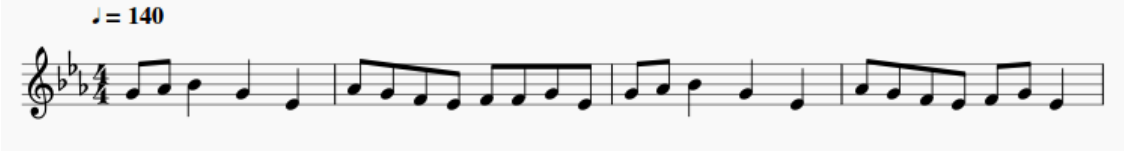


Figura D.33: Captura de pantalla de la partitura MIDI **Font:** Pròpia

Comentari: Partitura reconeguda al 100 %. Les notes de la quarta octava les ha reconegut totes.

Gegant del Pi en Solb Major

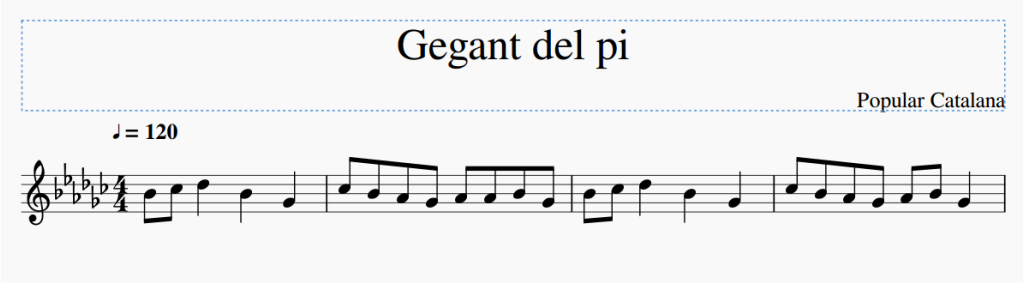


Figura D.34: Partitura Original **Font:** Pròpia

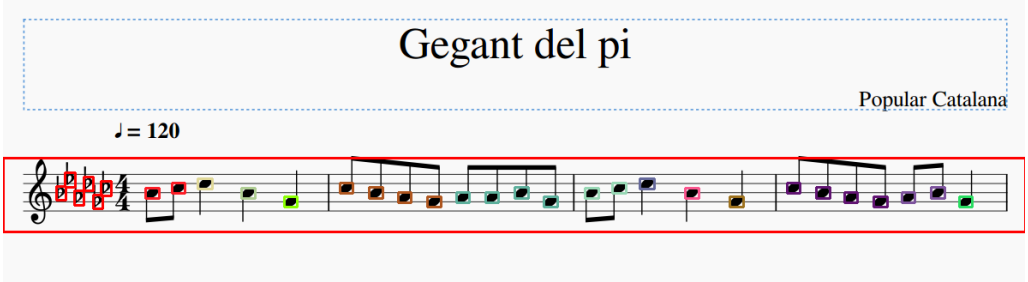


Figura D.35: Partitura amb els patrons reconeguts dibuixats **Font:** Pròpia

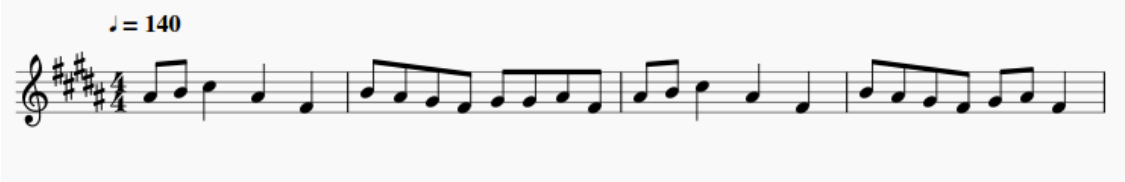


Figura D.36: Captura de pantalla de la partitura MIDI **Font:** Pròpia

Comentari: Partitura reconeguda al 100 %.

D.2. Partitures extretes d'Internet

Dins la fosca

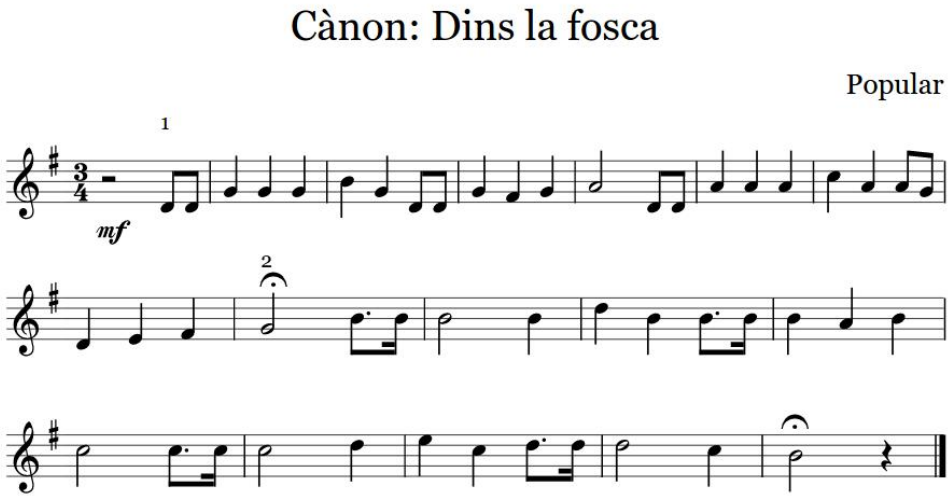


Figura D.37: Partitura Original **Font:** sites.google.com/site/wikimusiquem/recursos-tic/flauta/partitures

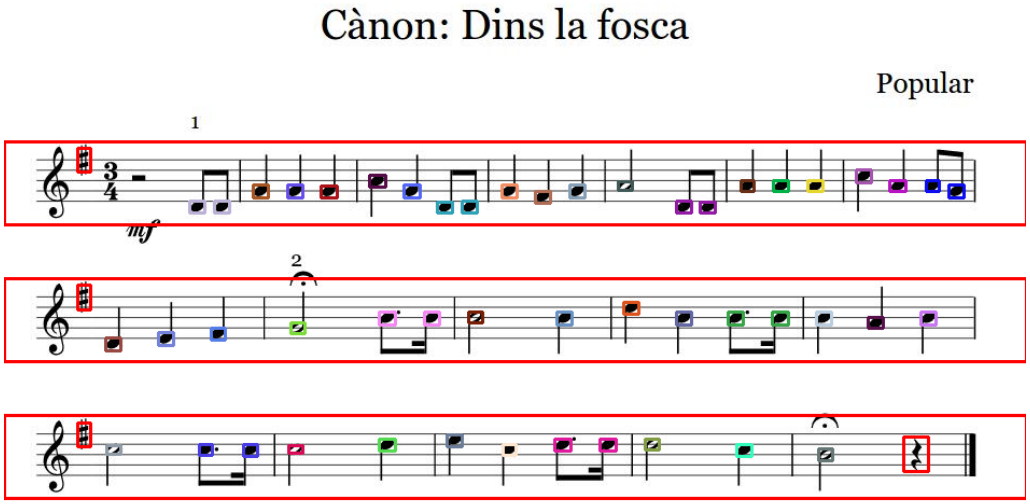


Figura D.38: Partitura amb els patrons reconeguts dibuixats **Font:** Pròpia

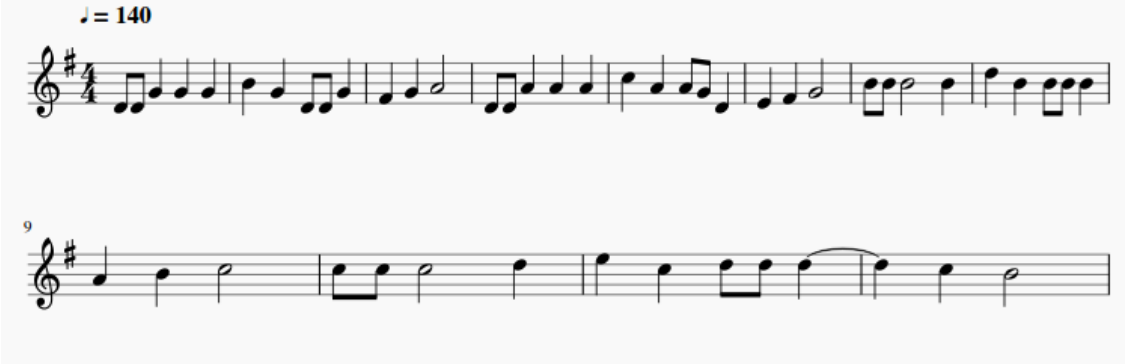


Figura D.39: Captura de pantalla de la partitura MIDI **Font:** Pròpia

Comentari: Partitura reconeguda al 100 %. Les semicorxeres que hi ha a la partitura, segons l'estàndard les reconeix com a corxeres. Per tant, ho ha fet bé. Els silencis també els ha reconegut però ignorat al generar el fitxer MIDI.

Moltes Felicitats

Figura D.40: Partitura Original **Font:**
sites.google.com/site/wikimusiquem/recursos-tic/flauta/partitures

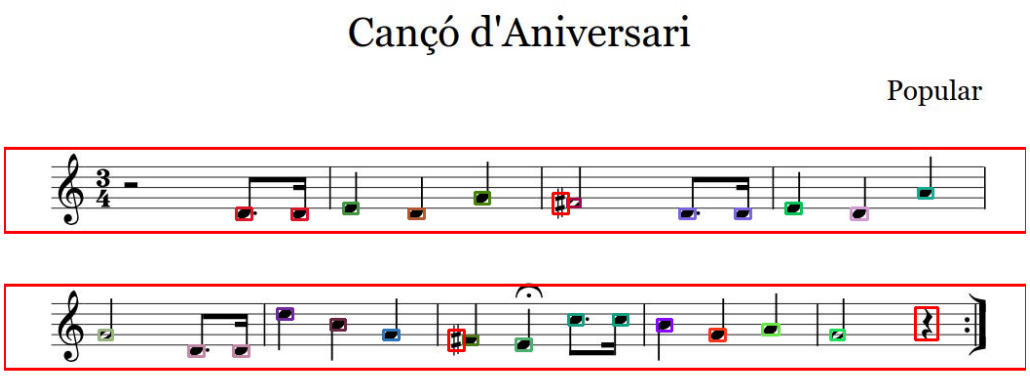


Figura D.41: Partitura amb els patrons reconeguts dibuixats **Font:** Pròpia

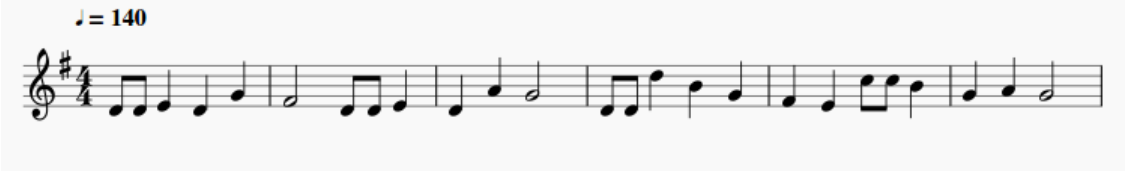


Figura D.42: Captura de pantalla de la partitura MIDI **Font:** Pròpia

Comentari: Partitura reconeguda al 100 %. Les semicorxeres que hi ha a la partitura, segons l'estàndard les reconeix com a corxeres. Per tant, ho ha fet bé. Els silencis també els ha reconegut però ignorat al generar el fitxer MIDI.

Tres i tres

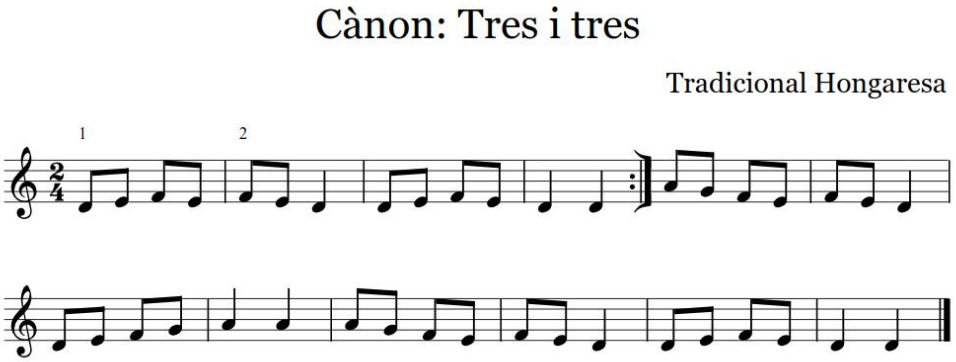


Figura D.43: Partitura Original **Font:**
sites.google.com/site/wikimusiquem/recursos-tic/flauta/partitures

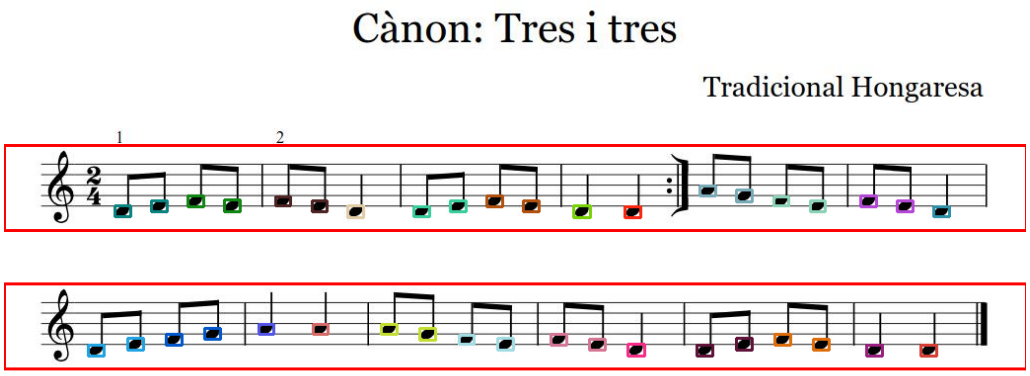


Figura D.44: Partitura amb els patrons reconeguts dibuixats **Font:** Pròpia

Figura D.45: Captura de pantalla de la partitura MIDI **Font:** Pròpia

Comentari: Partitura reconeguda al 100 %. La repetició no la fa perquè no la reconeix (segons estàndard).

El Matí

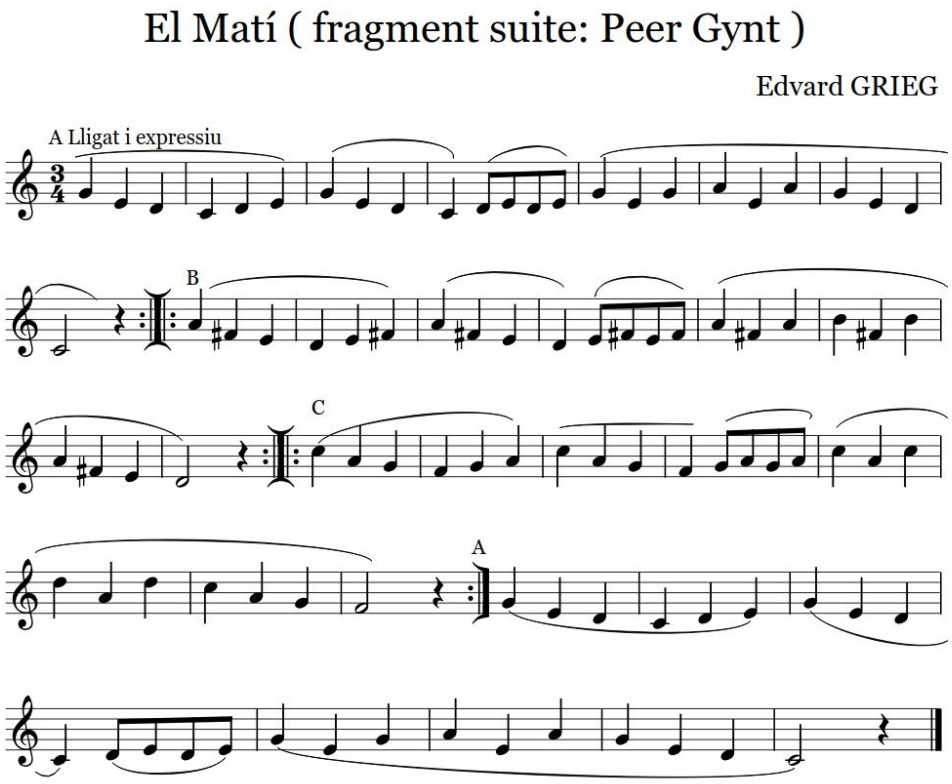


Figura D.46: Partitura Original **Font:**
sites.google.com/site/wikimusiquem/recursos-tic/flauta/partitures

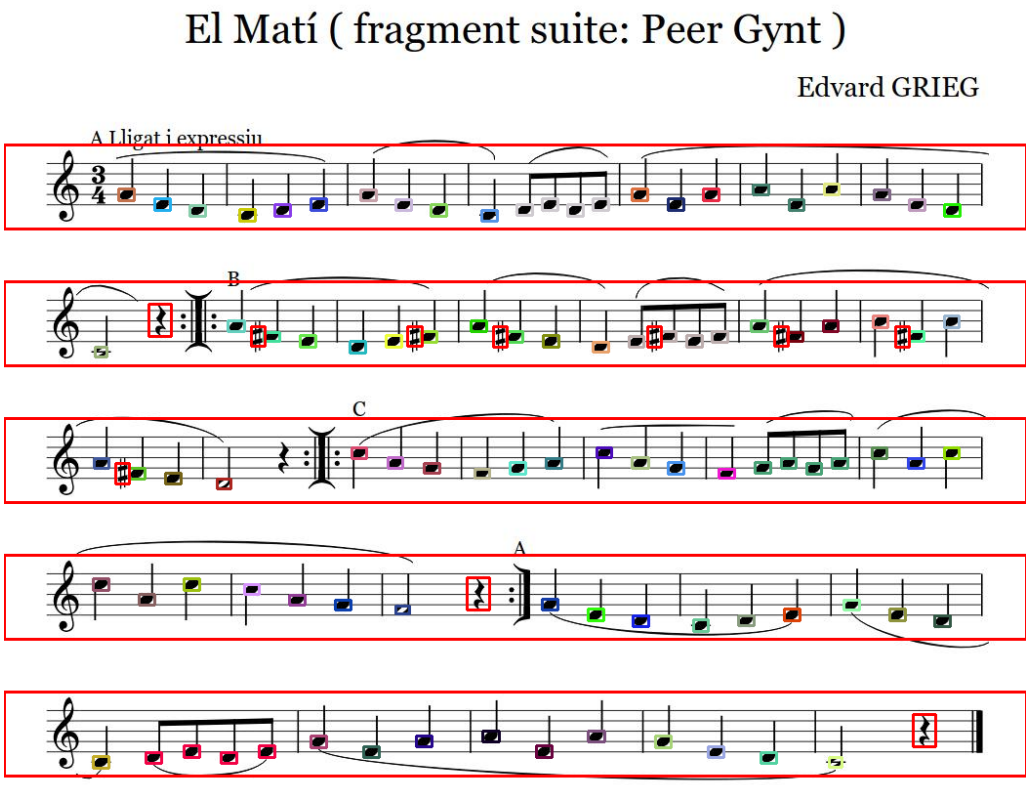


Figura D.47: Partitura amb els patrons reconeguts dibuixats **Font:** Pròpia

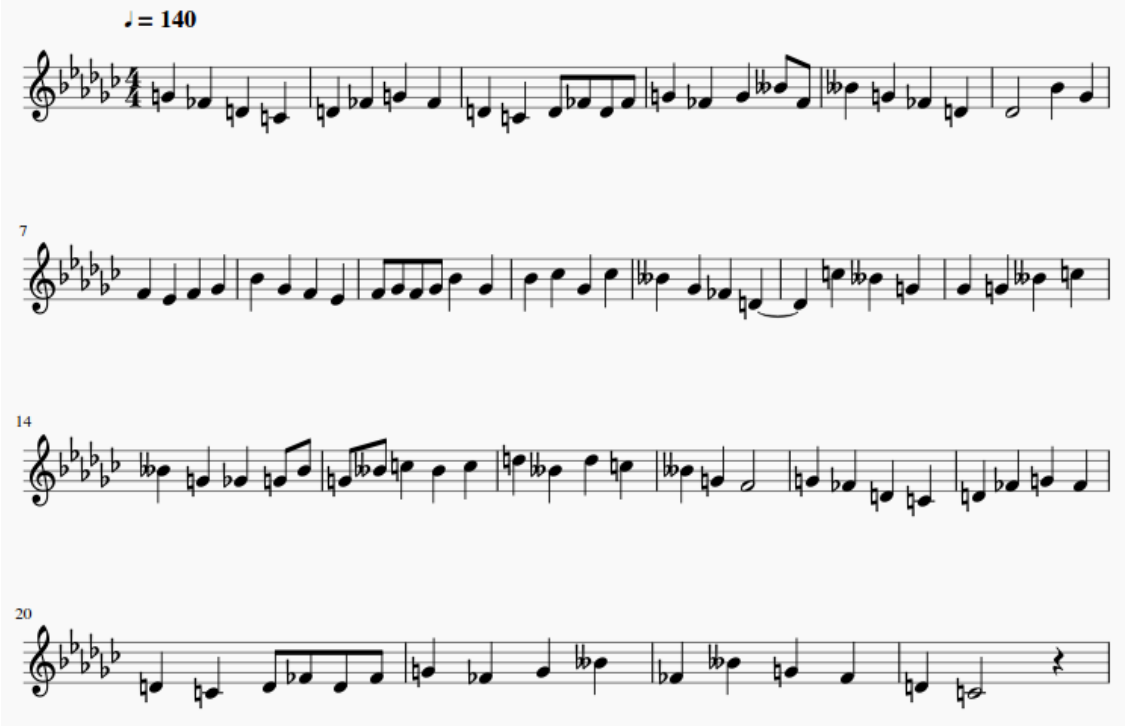


Figura D.48: Captura de pantalla de la partitura MIDI **Font:** Pròpia

Comentari: Partitura reconeguda al 92,26 %. Ha confós notes sobretot a la segona línia de la partitura. A simple vista és difícil de veure, pels motius que he explicat a l'inici del capítol, una partitura pot sonar igual però estar escrita diferent.

Himne de l'Alegria

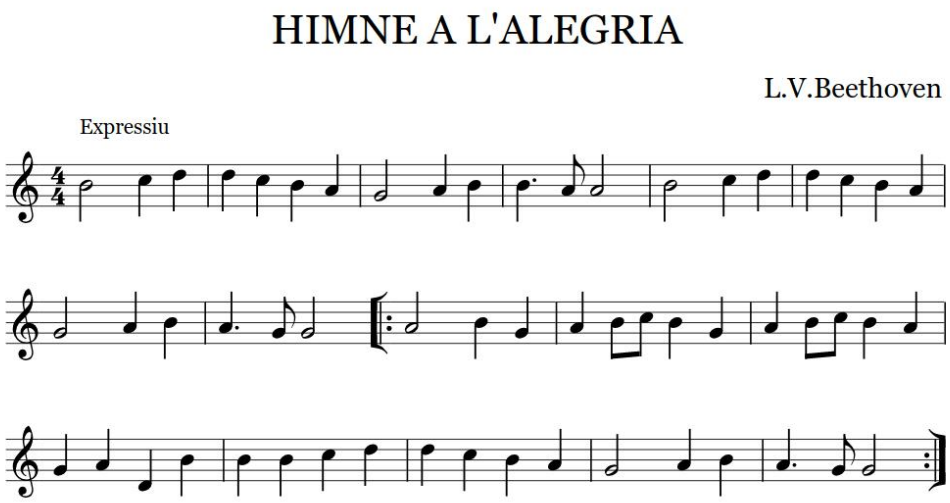


Figura D.49: Partitura Original **Font:** sites.google.com/site/wikimusiquem/recursos-tic/flauta/partitures

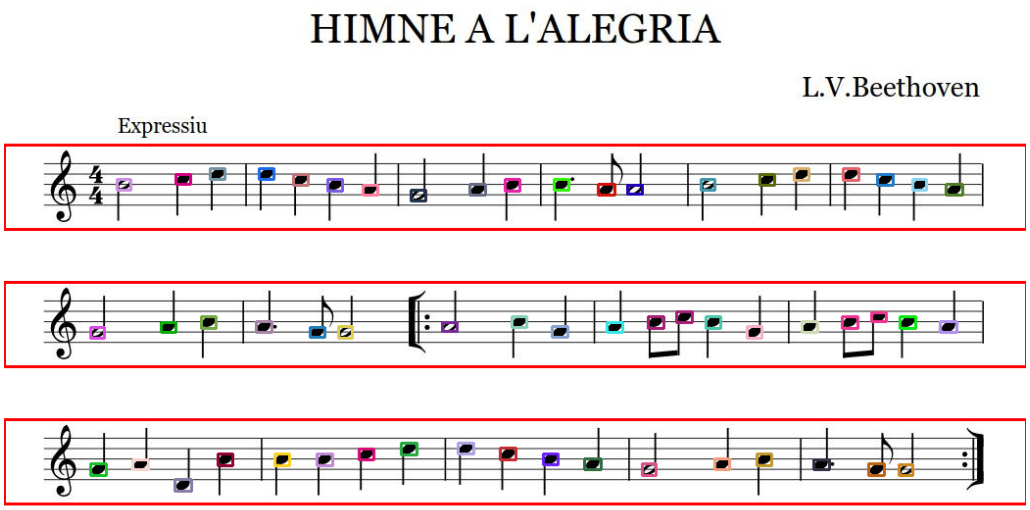


Figura D.50: Partitura amb els patrons reconeguts dibuixats **Font:** Pròpia

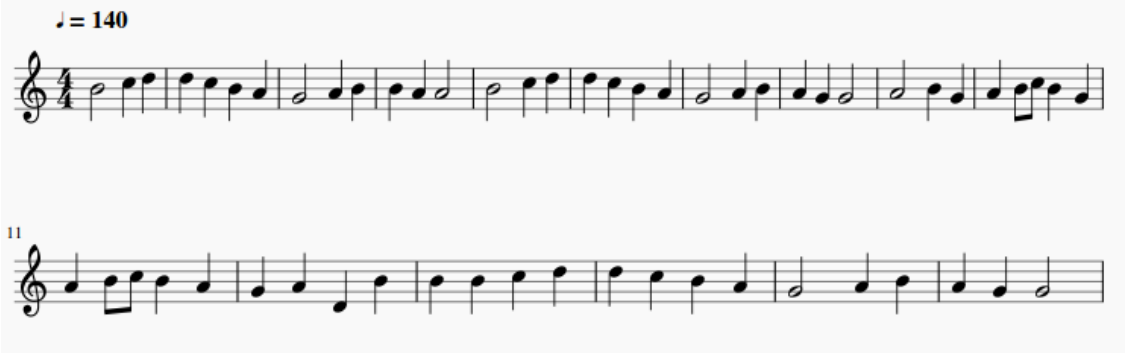


Figura D.51: Captura de pantalla de la partitura MIDI **Font:** Pròpia

Comentari: Partitura reconeguda al 100 %. Les negres amb punt les converteix a negres i les corxeres simples també a negres (segons estàndard). Tampoc fa repeticions.

Hora dels adéus

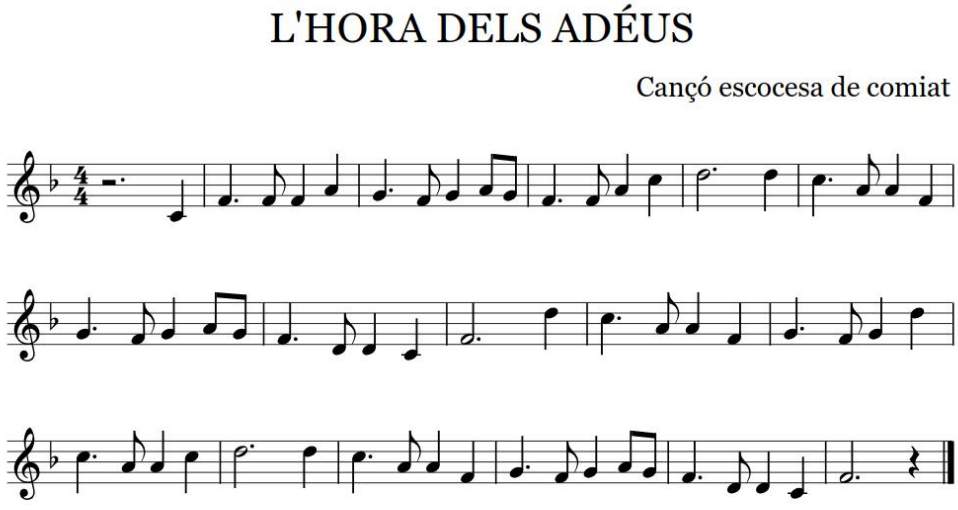


Figura D.52: Partitura Original **Font:**
sites.google.com/site/wikimusiquem/recursos-tic/flauta/partitures

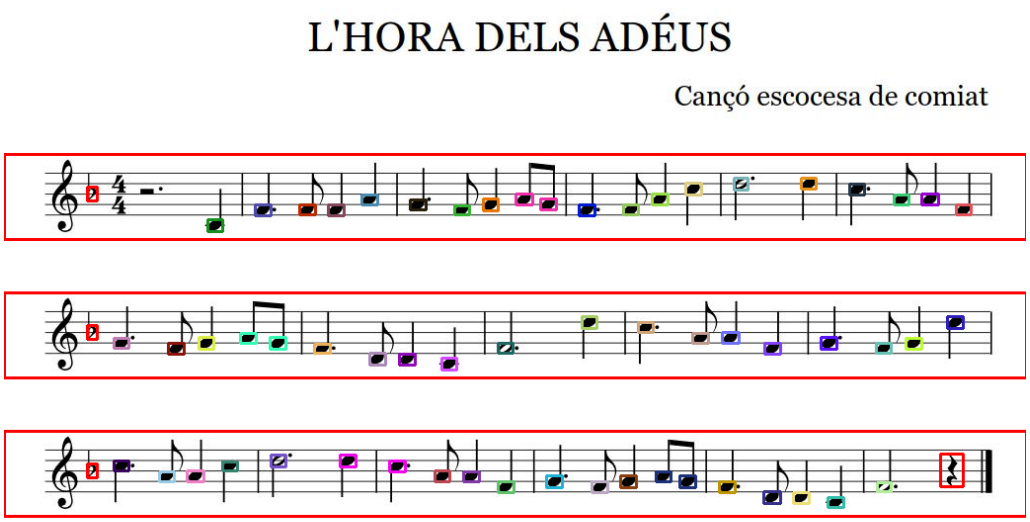


Figura D.53: Partitura amb els patrons reconeguts dibuixats **Font:** Pròpia

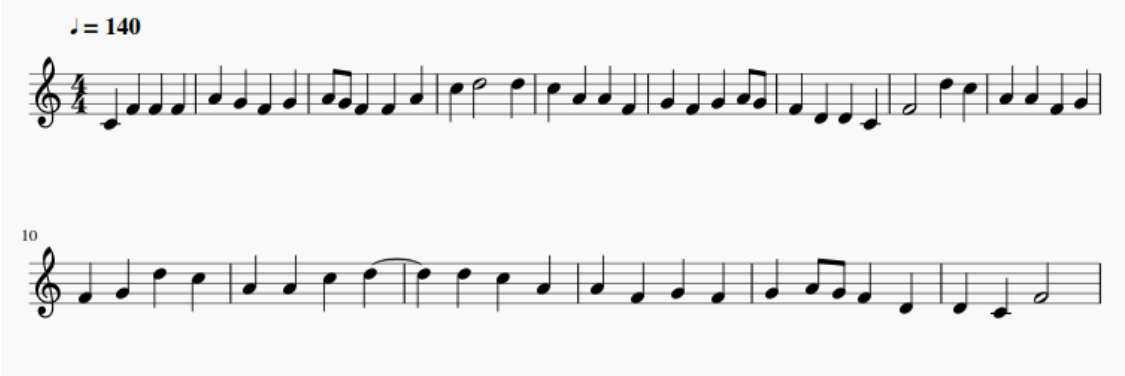


Figura D.54: Captura de pantalla de la partitura MIDI **Font:** Pròpia

Comentari: Partitura reconeguda al 100 %. Les notes que tenen un puntat al costat del cap, ignora el puntet. Les corxeres simples les converteix a negres a negres i els silencis els ignora (tot segons els paràmetres de l'estàndard).

Oh Susana!

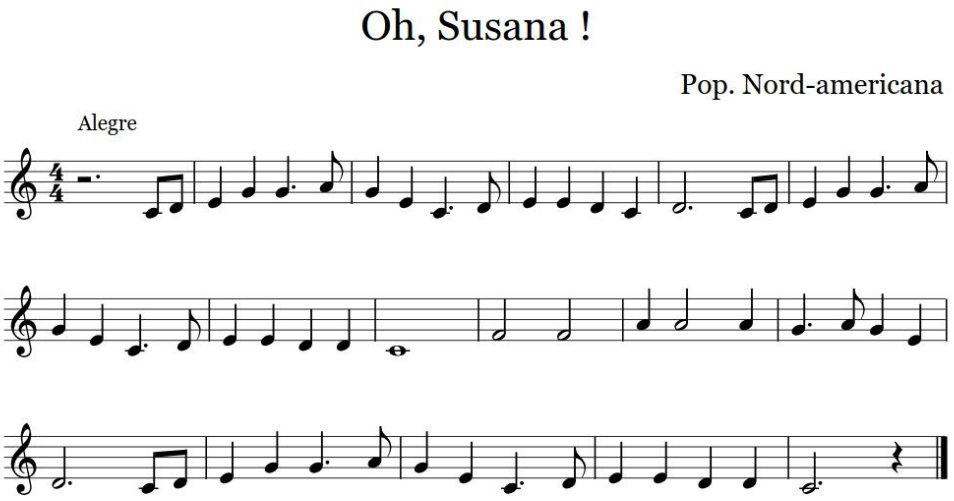


Figura D.55: Partitura Original **Font:** sites.google.com/site/wikimusiquem/recursos-tic/flauta/partitures

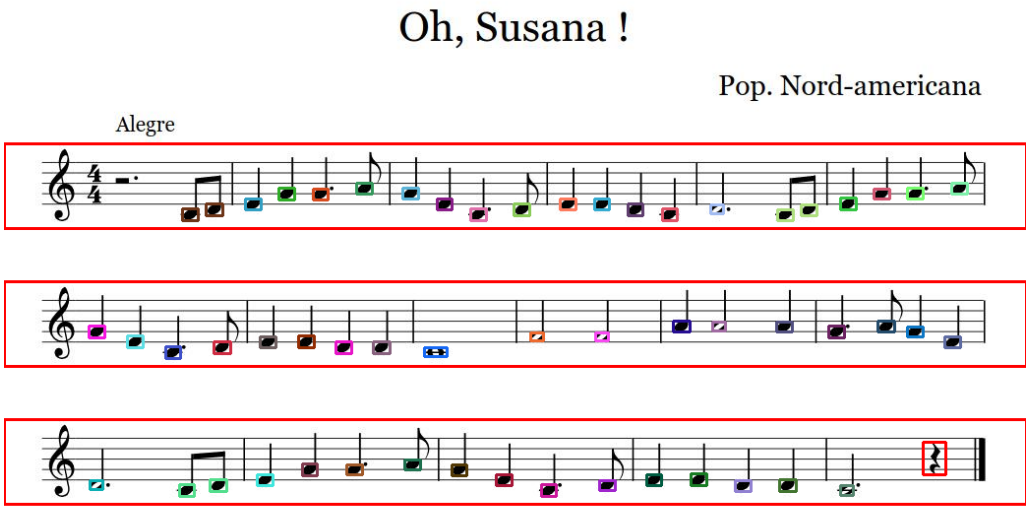


Figura D.56: Partitura amb els patrons reconeguts dibuixats **Font:** Pròpia

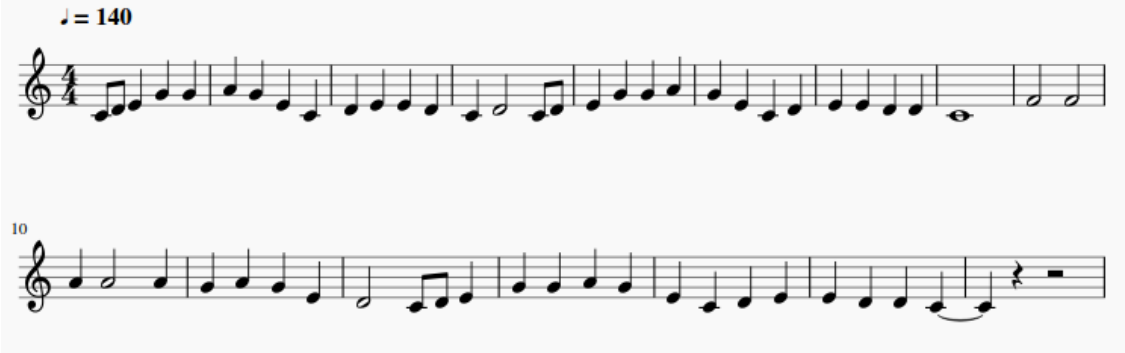


Figura D.57: Captura de pantalla de la partitura MIDI **Font:** Pròpia

Comentari: Partitura reconeguda al 100 %. Les notes que tenen un puntat al costat del cap, ignora el puntet. Les corxeres simples les converteix a negres a negres i els silencis els ignora (tot segons els paràmetres de l'estàndard).

Bob Dylan - Escolta-ho en el vent

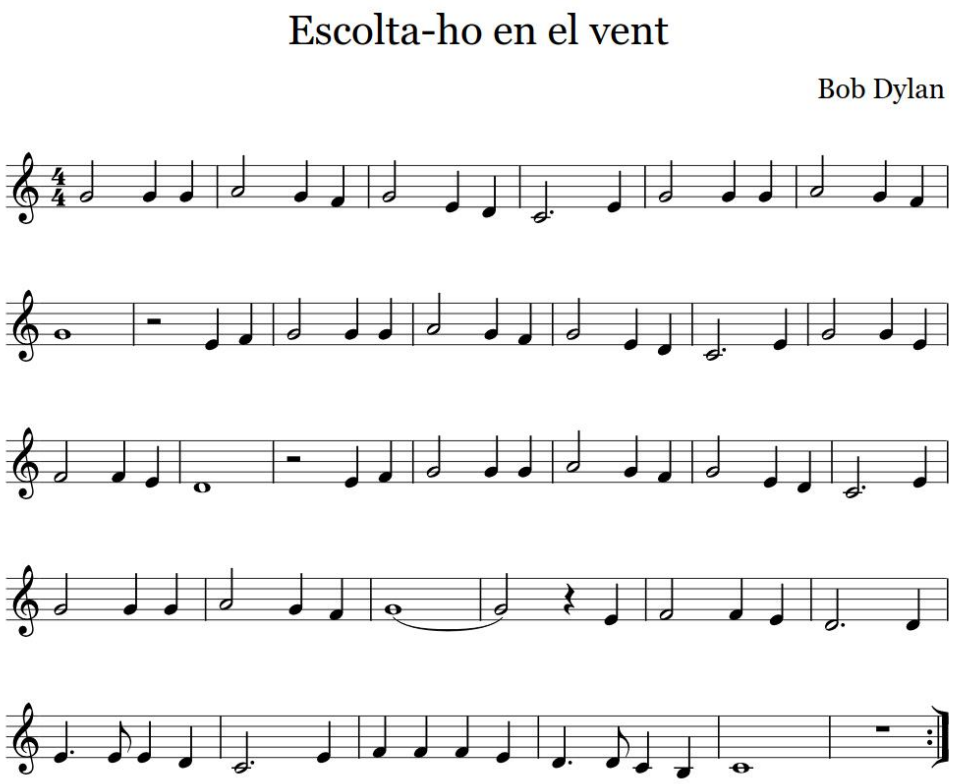


Figura D.58: Partitura Original **Font:** sites.google.com/site/wikimusiquem/recursos-tic/flauta/partitures

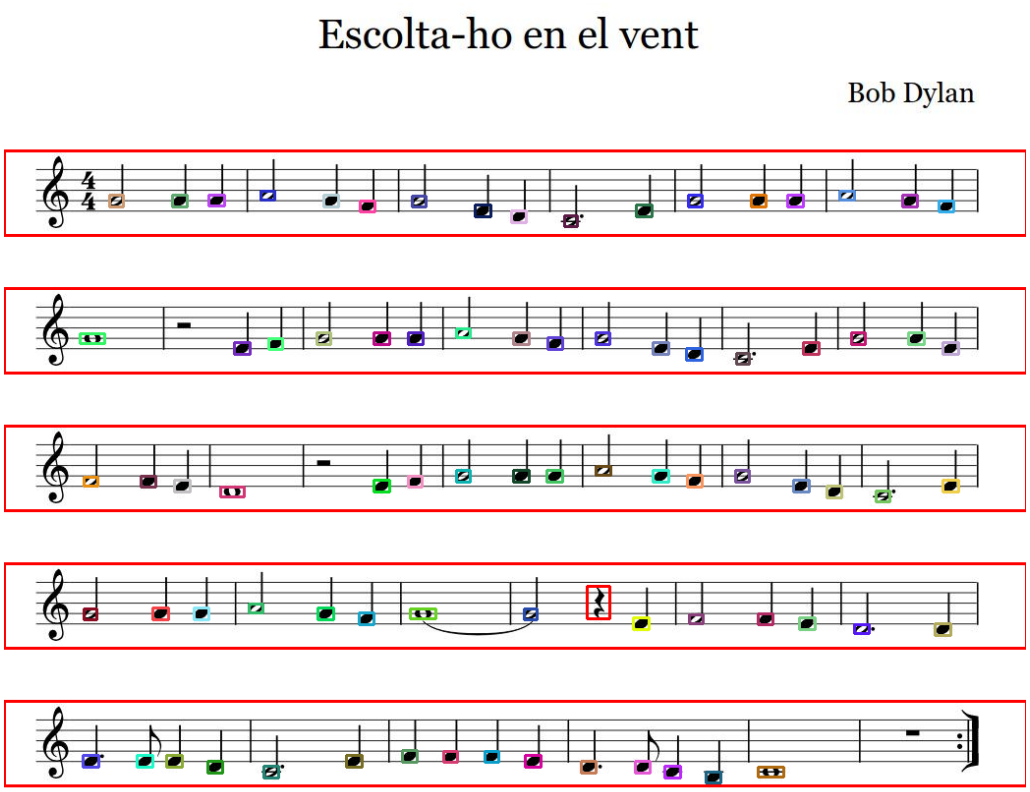


Figura D.59: Partitura amb els patrons reconeguts dibuixats **Font:** Pròpia

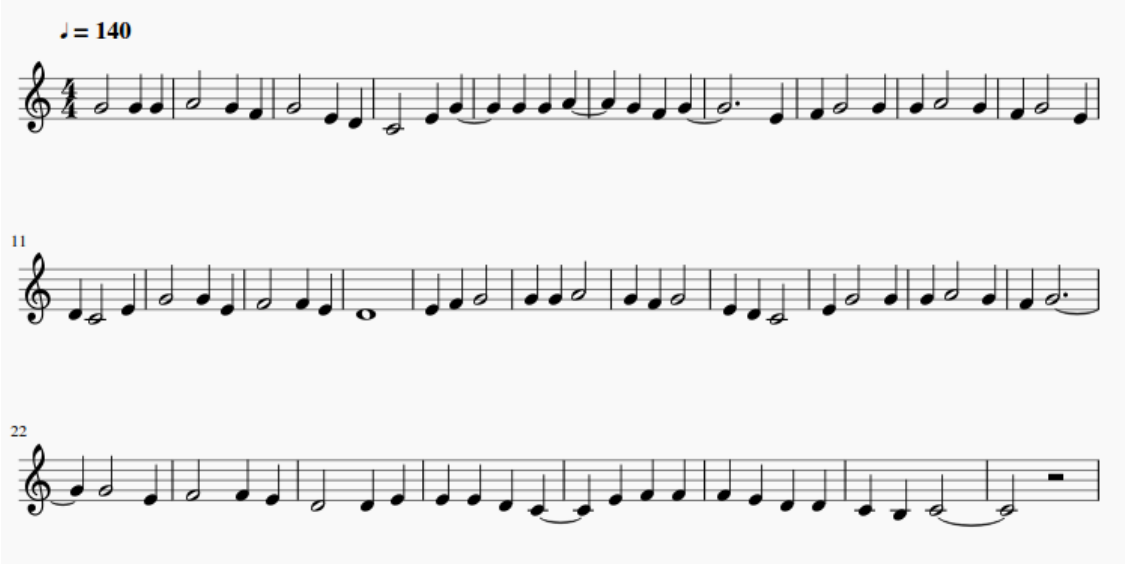


Figura D.60: Captura de pantalla de la partitura MIDI **Font:** Pròpia

Comentari: Partitura reconeguda al 100 %. Les blanques amb punt les converteix a blanques, les lligades, silencis i repeticions no les reconeix, (tot segons els paràmetres de l'estàndard).

Jingle Bell Rock

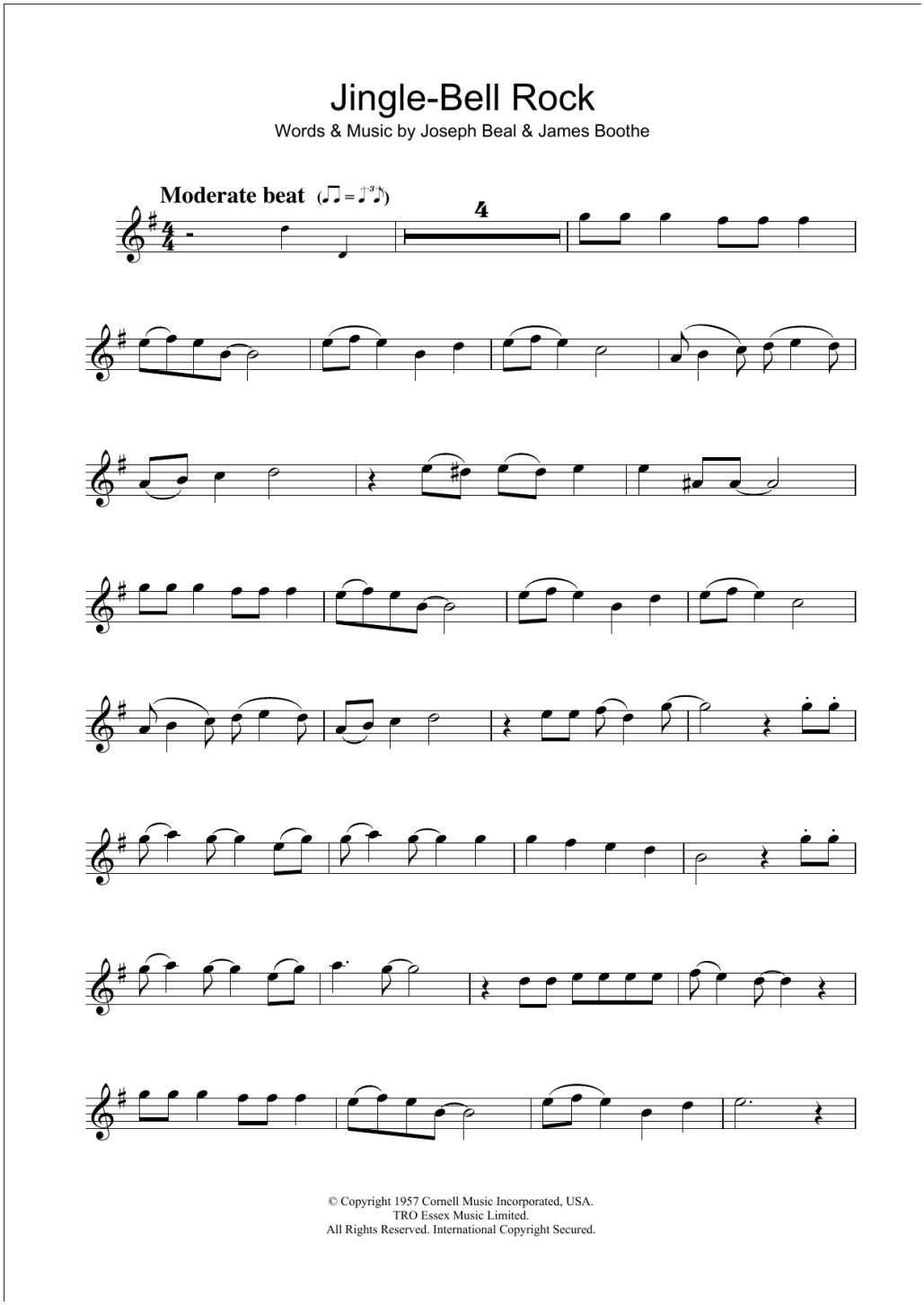


Figura D.61: Partitura Original **Font:** Google Imatges

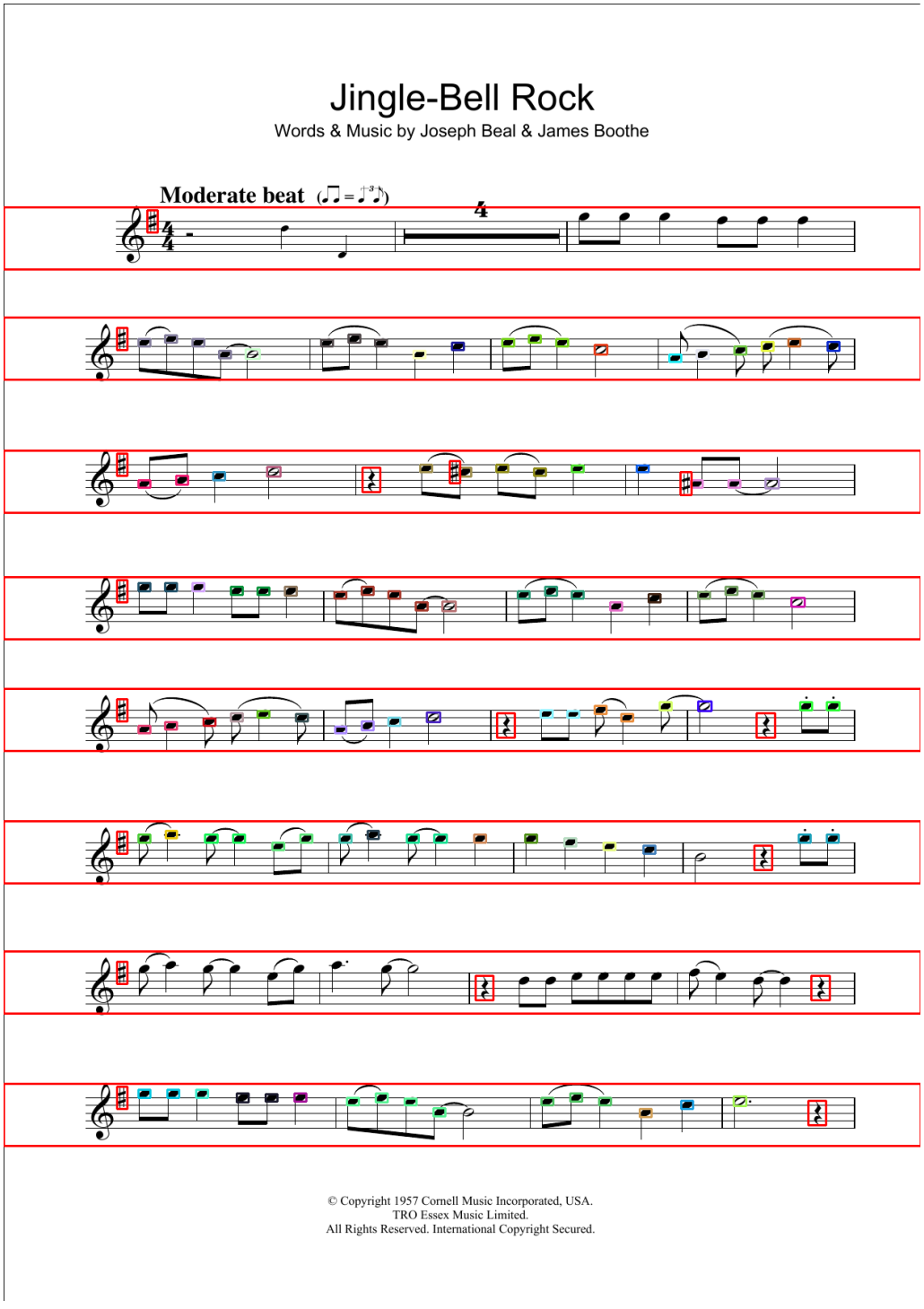


Figura D.62: Partitura amb els patrons reconeguts dibuixats **Font:** Pròpia

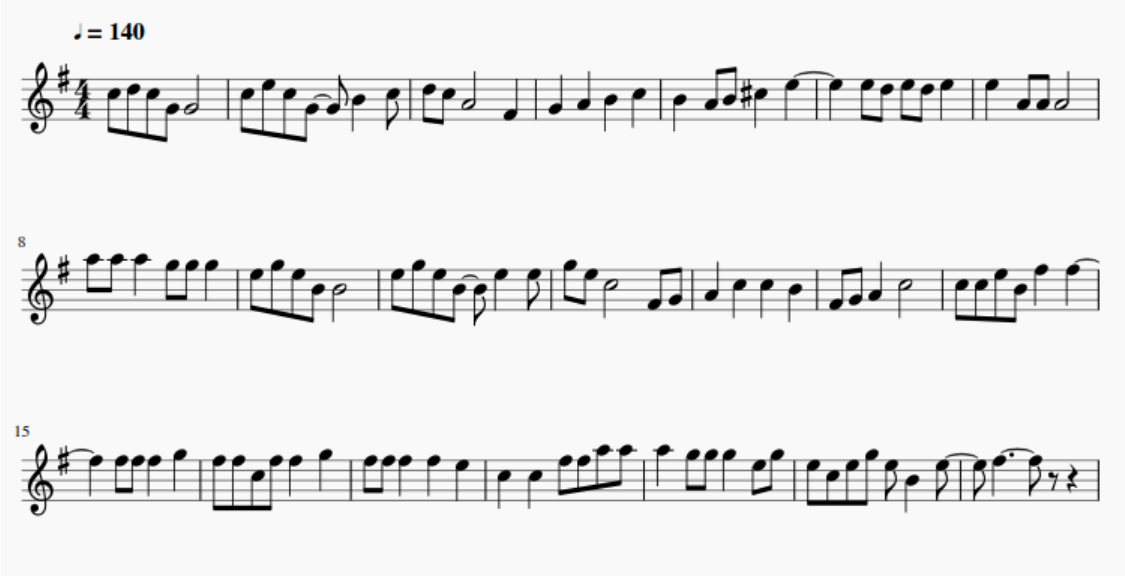


Figura D.63: Captura de pantalla de la partitura MIDI **Font:** Pròpia

Comentari: Partitura reconeguda al 29,87 %. Es pot veure com hi ha línies senceres de la partitura en què no ha reconegut els patrons de les notes. Això és perquè els valors d'escalat mínim i màxim per aquesta partitura no estan ben ajustats.

Scarborough Fair

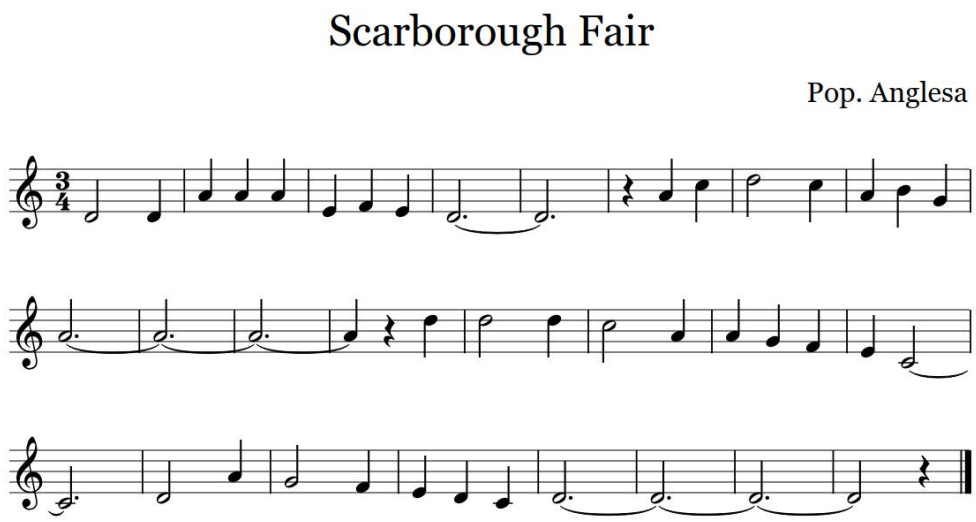


Figura D.64: Partitura Original **Font:**
sites.google.com/site/wikimusiquem/recursos-tic/flauta/partitures

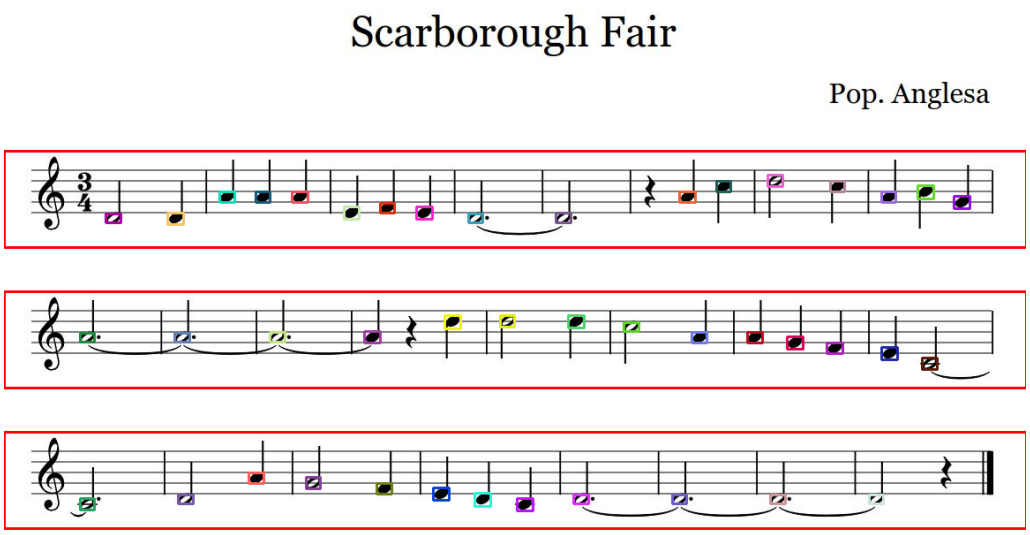


Figura D.65: Partitura amb els patrons reconeguts dibuixats **Font:** Pròpia

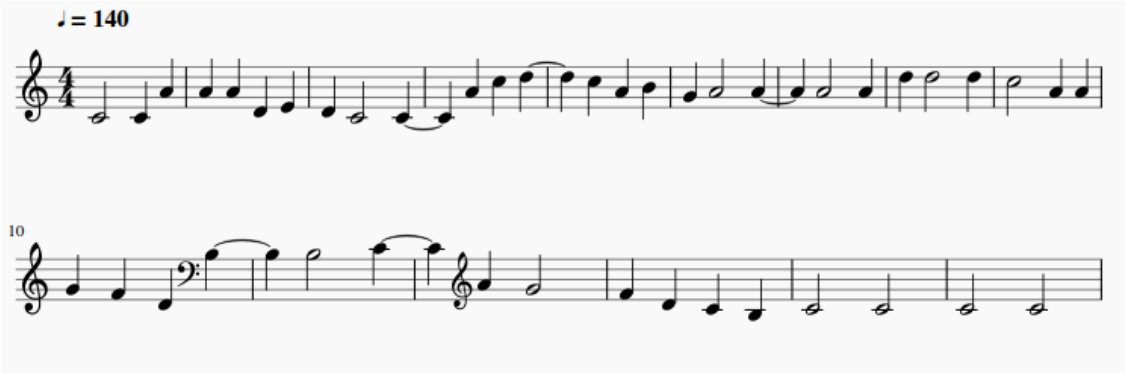


Figura D.66: Captura de pantalla de la partitura MIDI **Font:** Pròpia

Comentari: Partitura reconeguda al 83,72 %. No ha reconegut bé l'altura d'algunes notes (sobretot les més greus).

My Heart will go on - Titanic

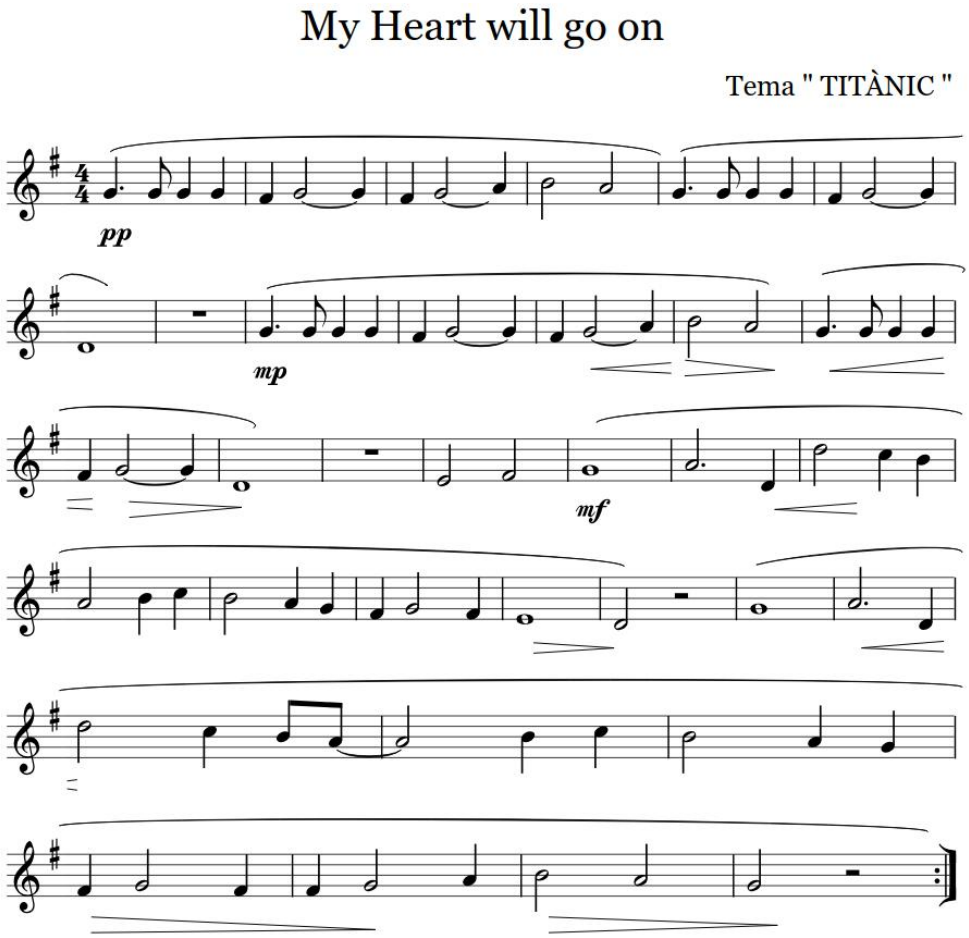


Figura D.67: Partitura Original **Font:**
sites.google.com/site/wikimusiquem/recursos-tic/flauta/partitures

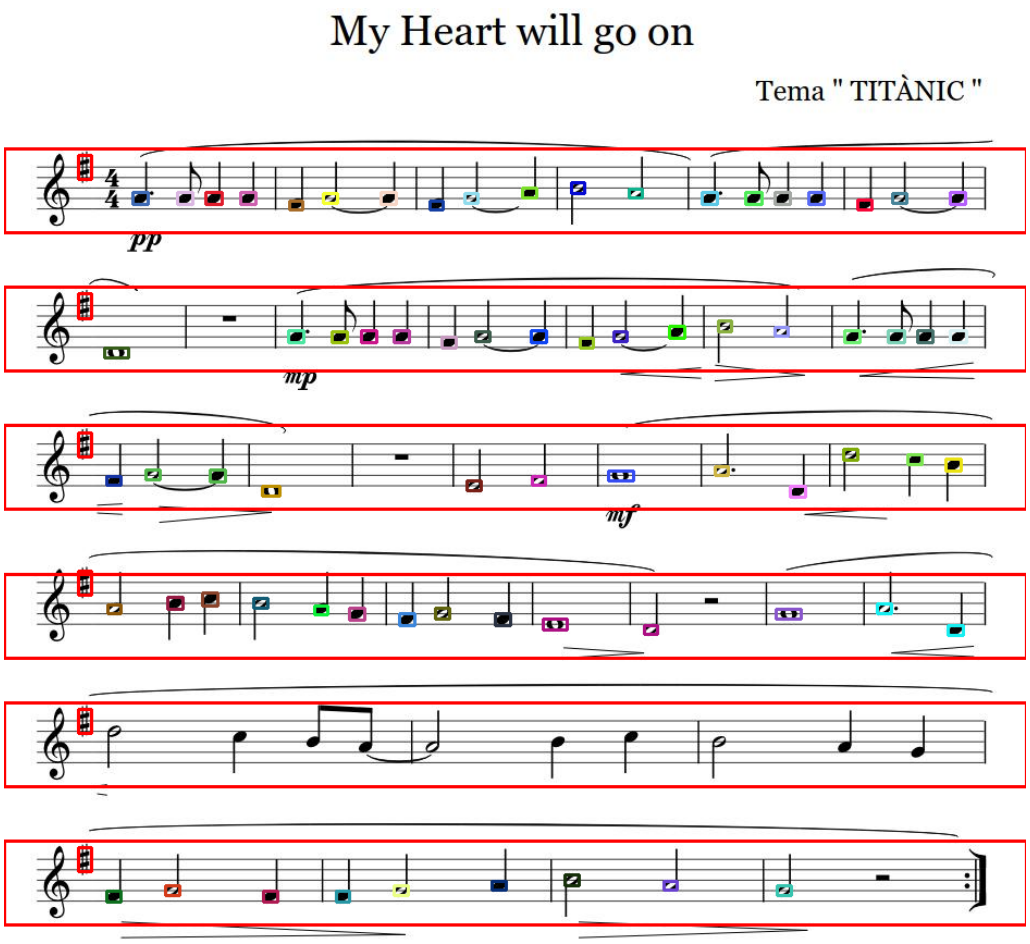


Figura D.68: Partitura amb els patrons reconeguts dibuixats **Font:** Pròpia

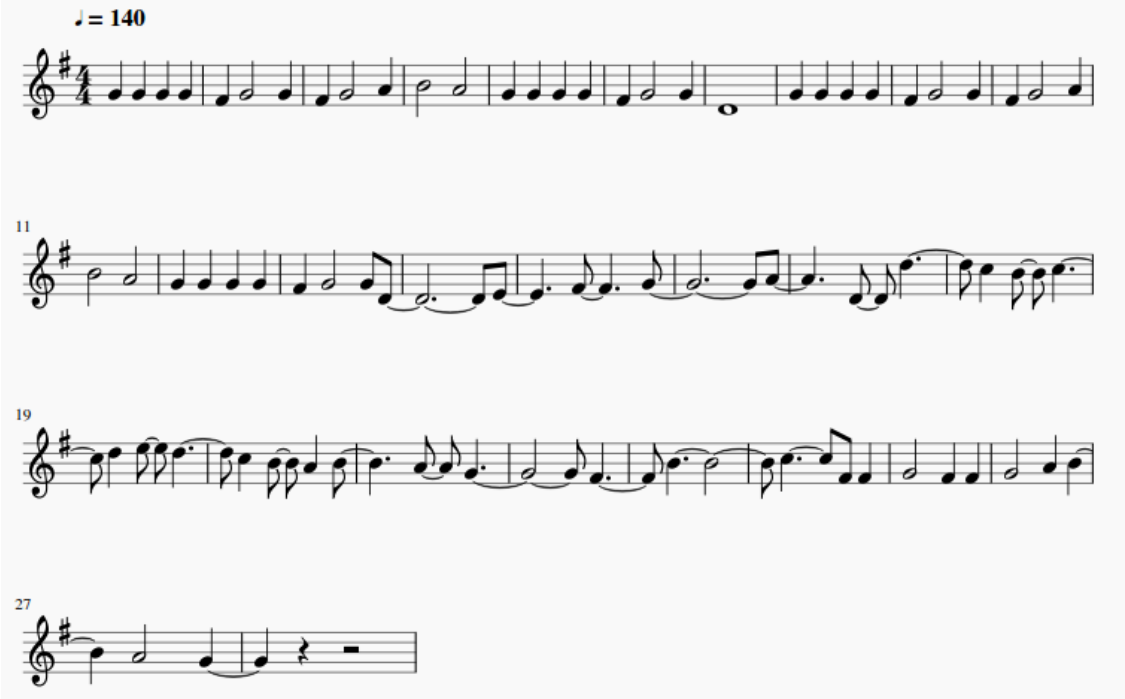


Figura D.69: Captura de pantalla de la partitura MIDI **Font:** Pròpia

Comentari: Partitura reconeguda al 79,48 %. Una línia de la partitura no l'ha reconegut i part d'una altra línia ha reconegut malament les notes. Les repeticions, lligades i dinàmiques no les reconeix (segons estàndard).

Baixant de la font del gat

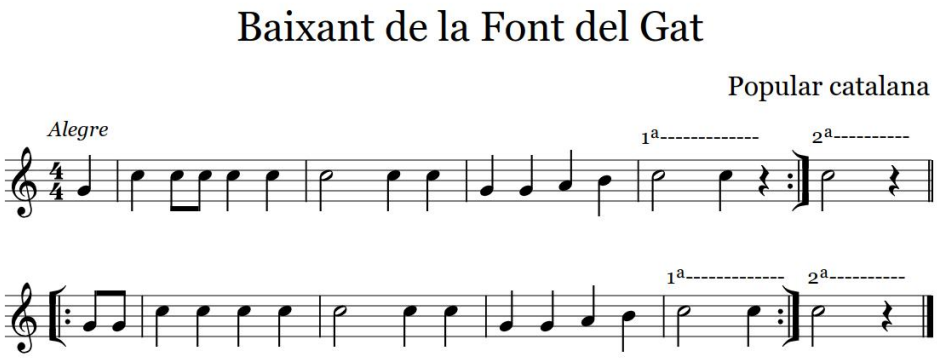


Figura D.70: Partitura Original **Font:**
sites.google.com/site/wikimusiquem/recursos-tic/flauta/partitures

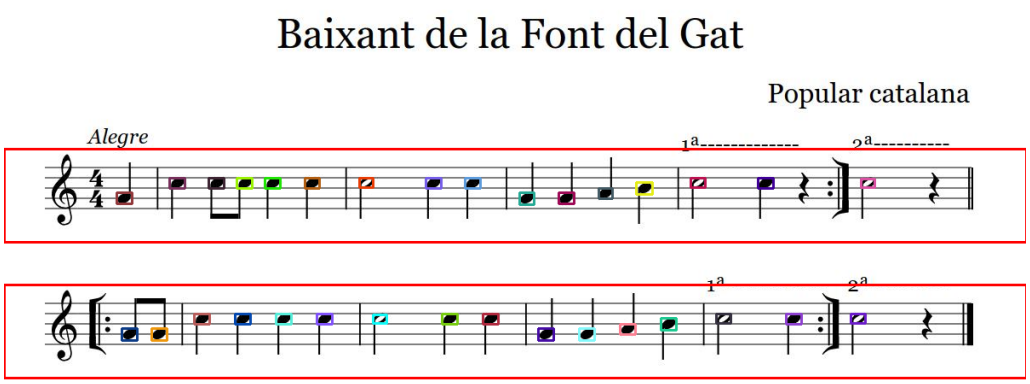


Figura D.71: Partitura amb els patrons reconeguts dibuixats **Font:** Pròpia

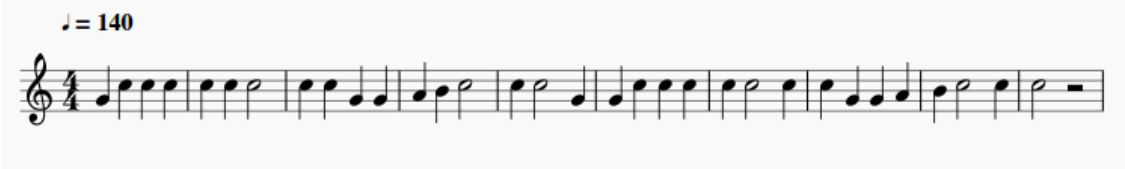


Figura D.72: Captura de pantalla de la partitura MIDI **Font:** Pròpia

Comentari: Partitura reconeguda al 93,75 %. Ha confós corxeres per negres.

Boig per tu - Sau

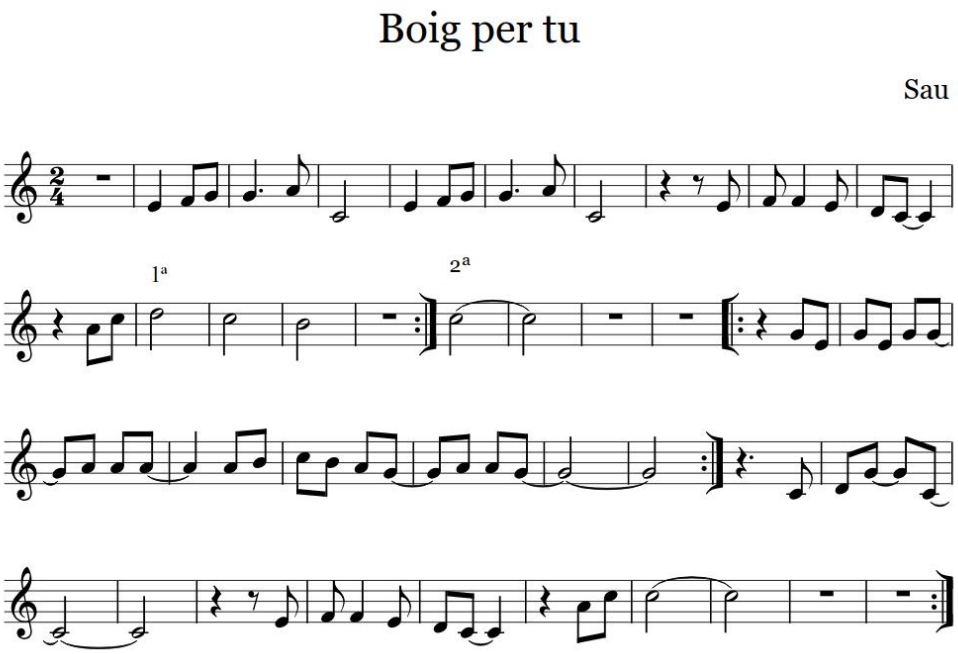


Figura D.73: Partitura Original **Font:** sites.google.com/site/wikimusiquem/recursos-tic/flauta/partitures

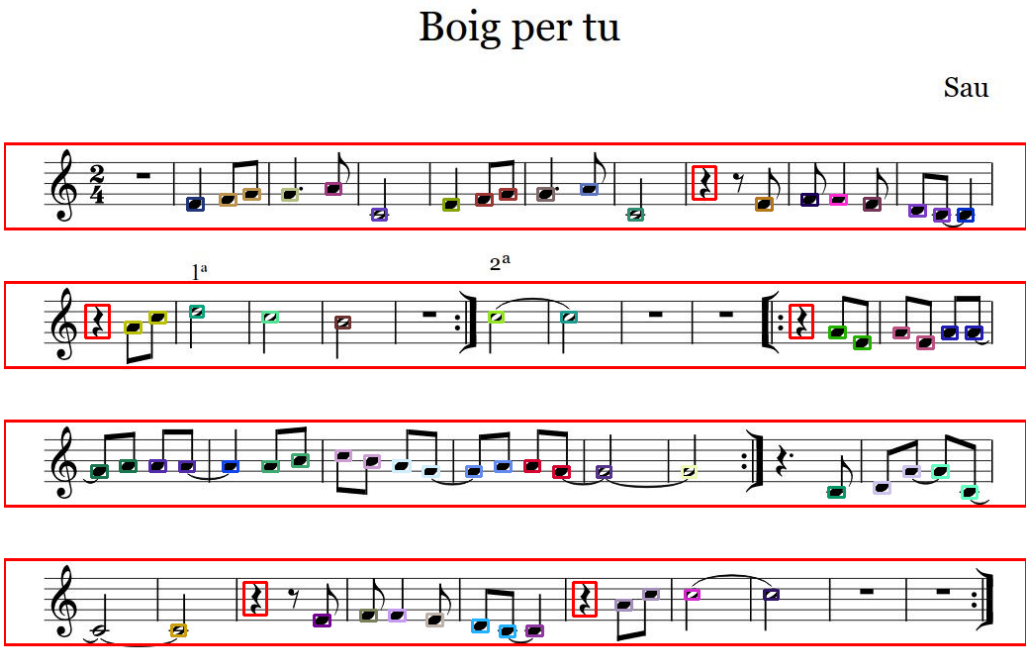


Figura D.74: Partitura amb els patrons reconeguts dibuixats **Font:** Pròpia

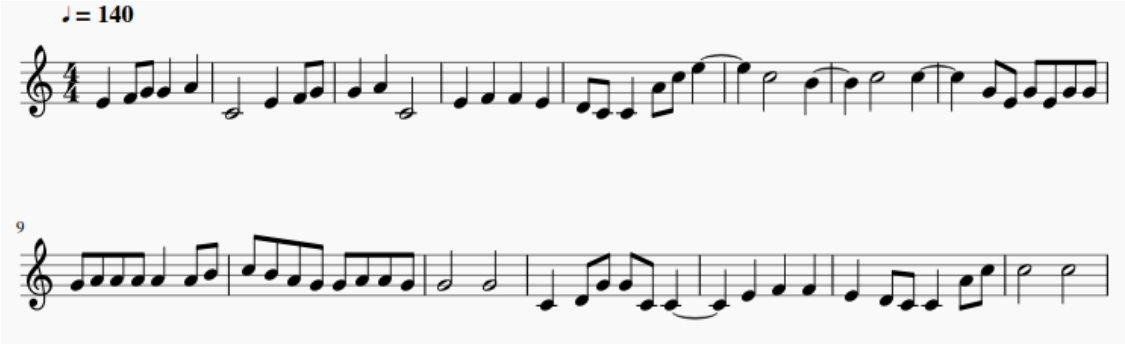


Figura D.75: Captura de pantalla de la partitura MIDI **Font:** Pròpia

Comentari: Partitura reconeguda al 99,25 %. Les lligades, silencis i repeticions no les reconeix. Les negres amb punt les passa a negres i les corxeres simples a negres. S'ha equivocat reconeixent la nota Re5 per Mi5.

Santa Nit

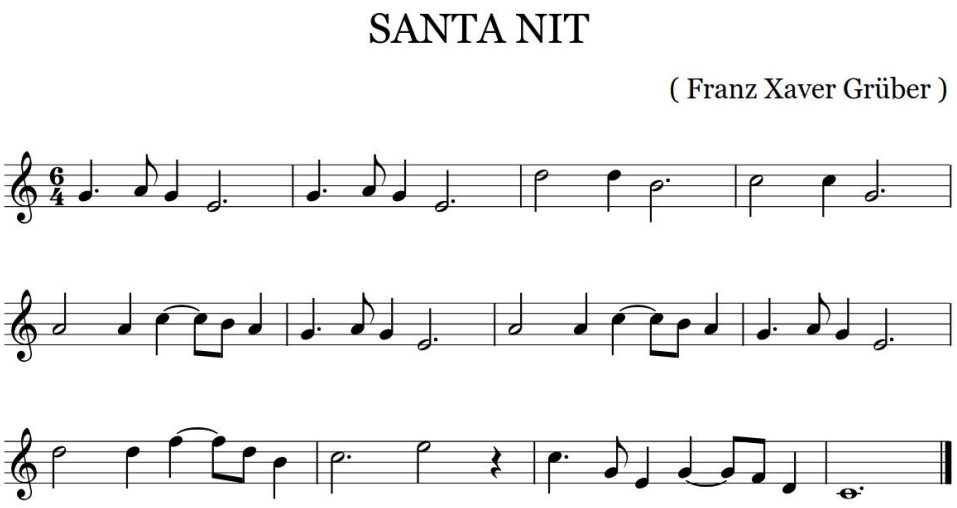


Figura D.76: Partitura Original **Font:**
sites.google.com/site/wikimusiquem/recursos-tic/flauta/partitures

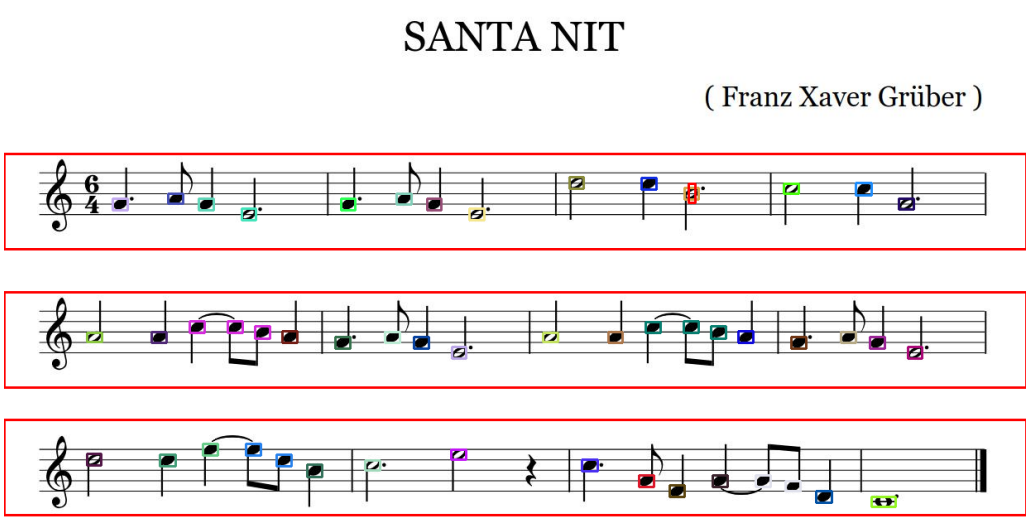


Figura D.77: Partitura amb els patrons reconeguts dibuixats **Font:** Pròpia

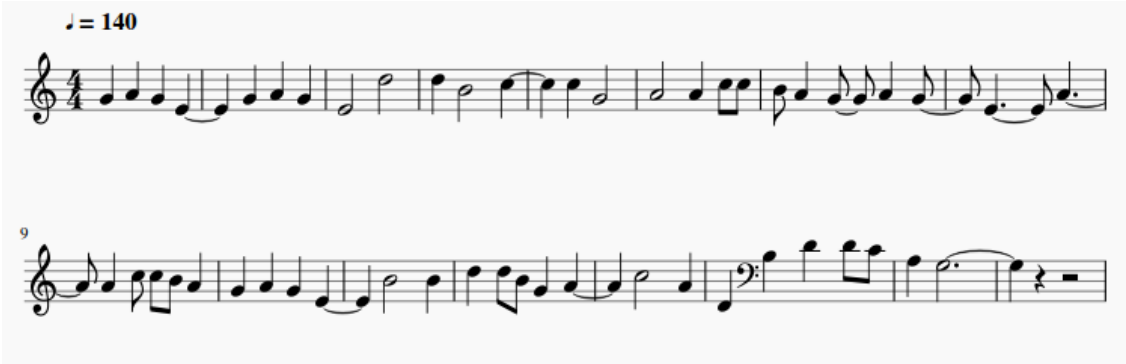


Figura D.78: Captura de pantalla de la partitura MIDI **Font:** Pròpia

Comentari: Partitura reconeguda al 80 %. Les notes amb un punt al costat del cap, ignora el punt. Les corxeres simples les converteix a negres i els silencis i les lligades les ignora, (segons estàndard).

No ha reconegut bé les notes de la última línia de la partitura.

D.3. Partitures escanejades d'un llibre de música

Quan les oques

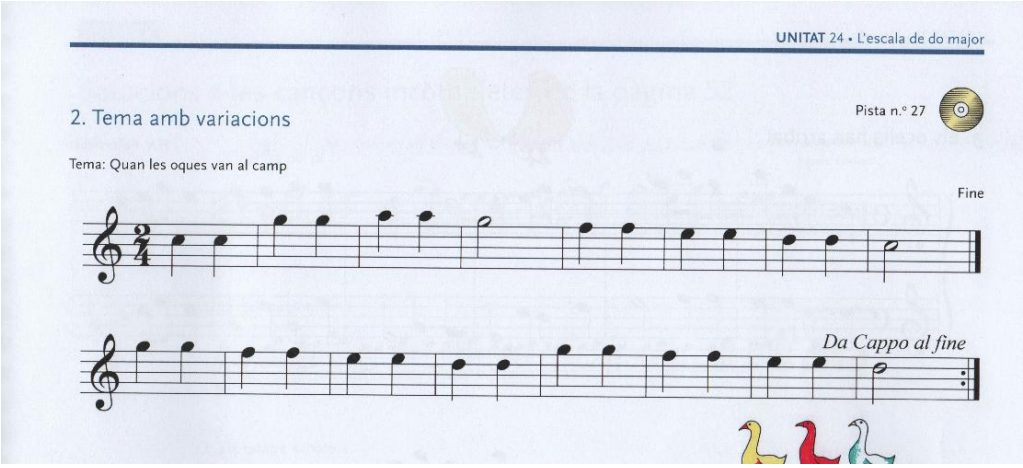


Figura D.79: Partitura Original
Font: Llibre "*Aprèn jugant amb la flauta travessera*", Volum 1

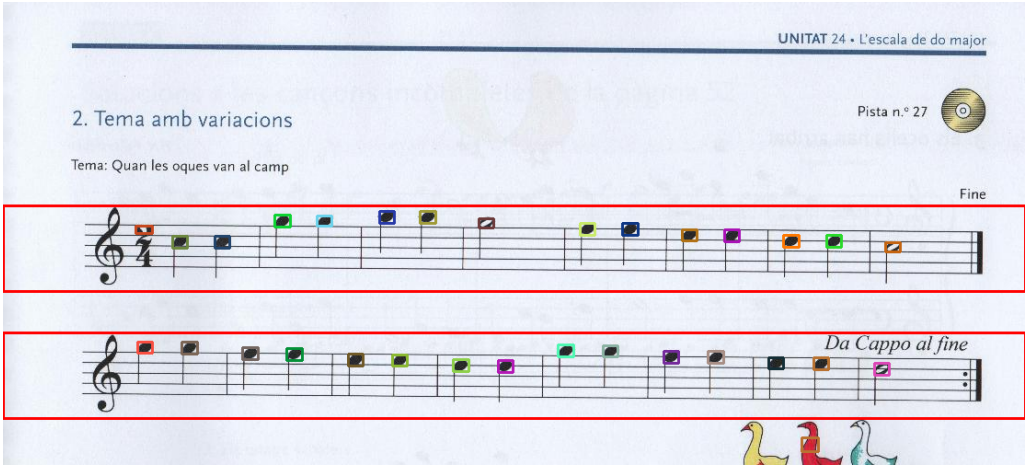


Figura D.80: Partitura amb els patrons reconeguts dibuixats **Font:** Pròpia

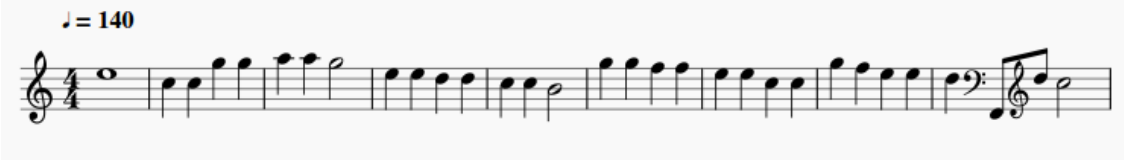


Figura D.81: Captura de pantalla de la partitura MIDI **Font:** Pròpia

Comentari: Partitura reconeguda al 70,69 %. El número 2 que indica el tipus de compàs inicial de la partitura l'ha reconegut com un Mi5 rodona. També, al estar la partitura torta, ha reconegut malament part de les notes.

Un, dos, ra

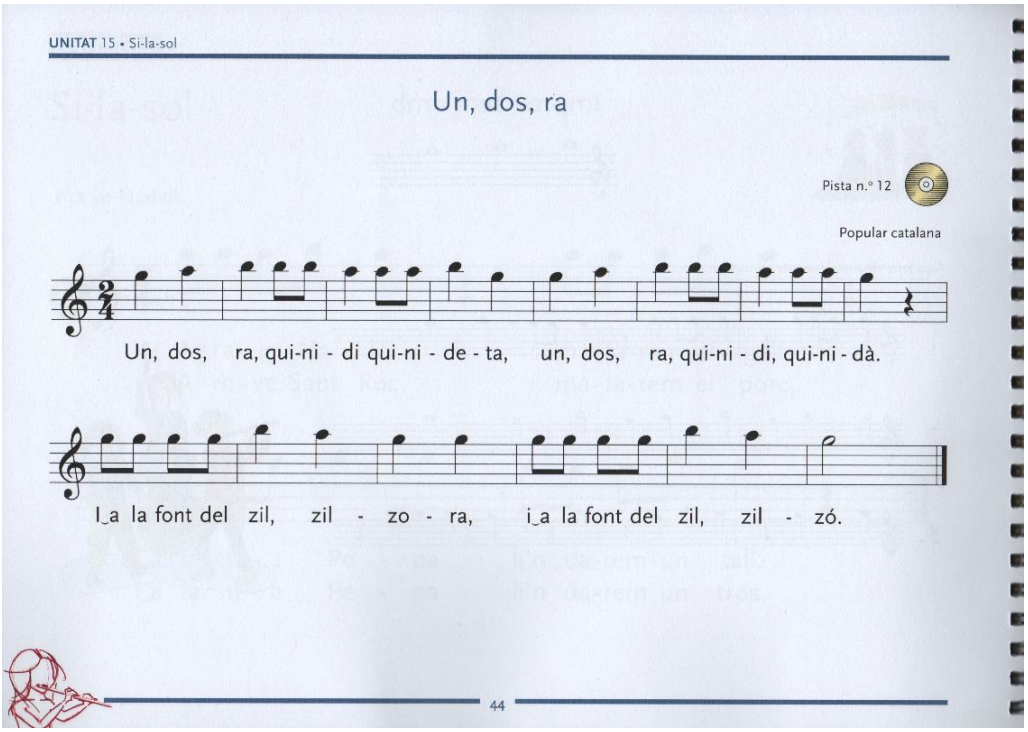


Figura D.82: Partitura Original
Font: Llibre "*Aprèn jugant amb la flauta travessera*", Volum 1

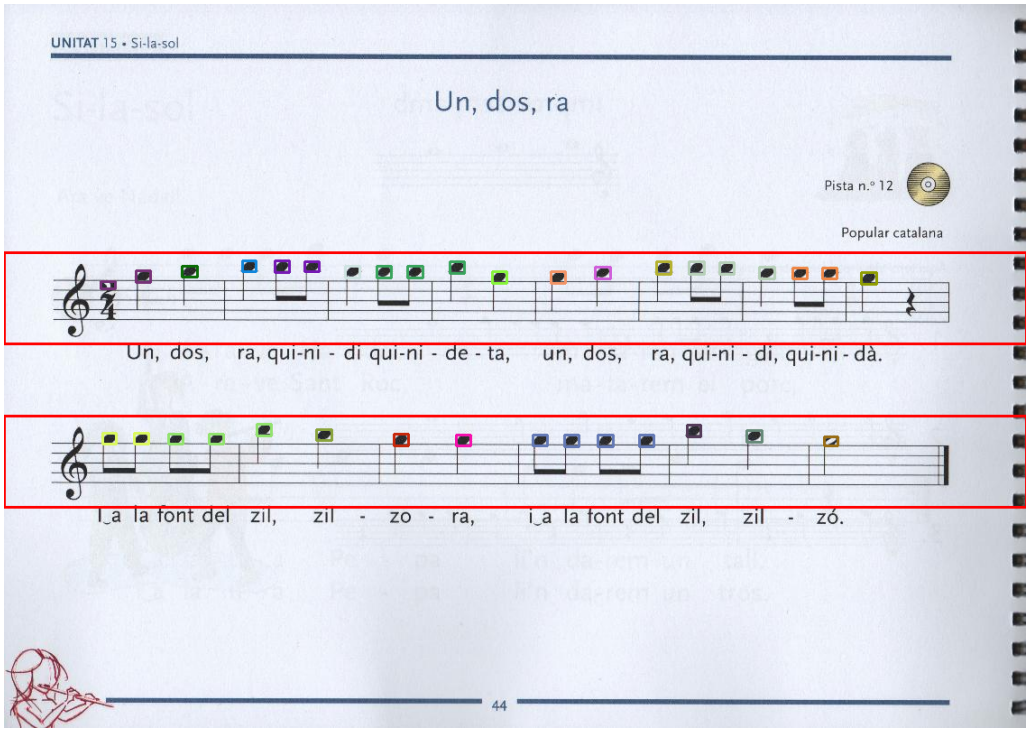


Figura D.83: Partitura amb els patrons reconeguts dibuixats **Font:** Pròpia

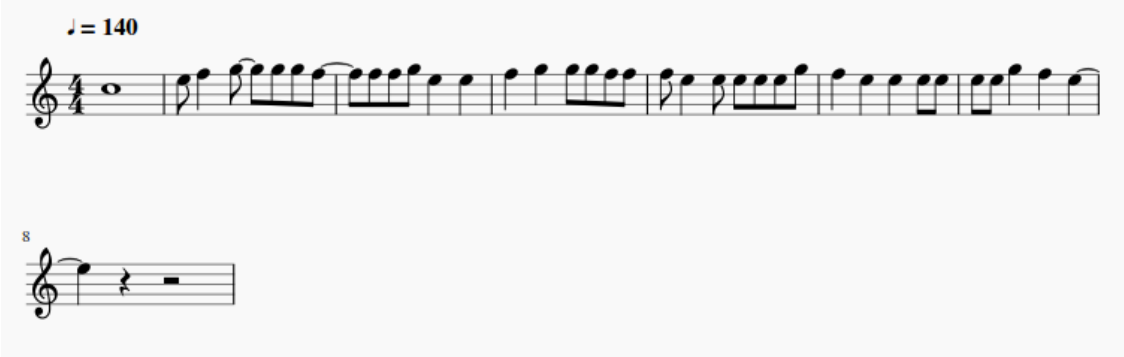


Figura D.84: Captura de pantalla de la partitura MIDI **Font:** Pròpia

Comentari: Partitura reconeguda al 42,65 %. El número 2 que indica el tipus de compàs inicial de la partitura l'ha reconegut com un Mi5 rodona. Al ser notes tan agudes no les ha reconegut bé. Les ha reconegut un interval d'una tercera major per sota.

Cançó del Picapedrer

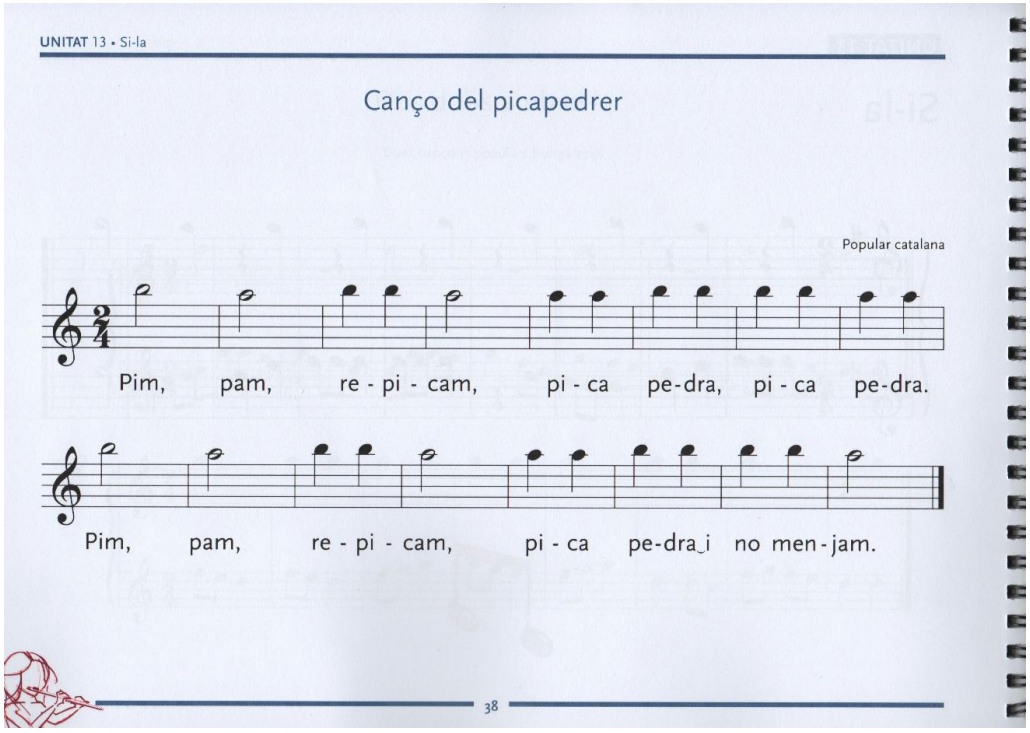


Figura D.85: Partitura Original
Font: Llibre "*Aprèn jugant amb la flauta travessera*", Volum 1

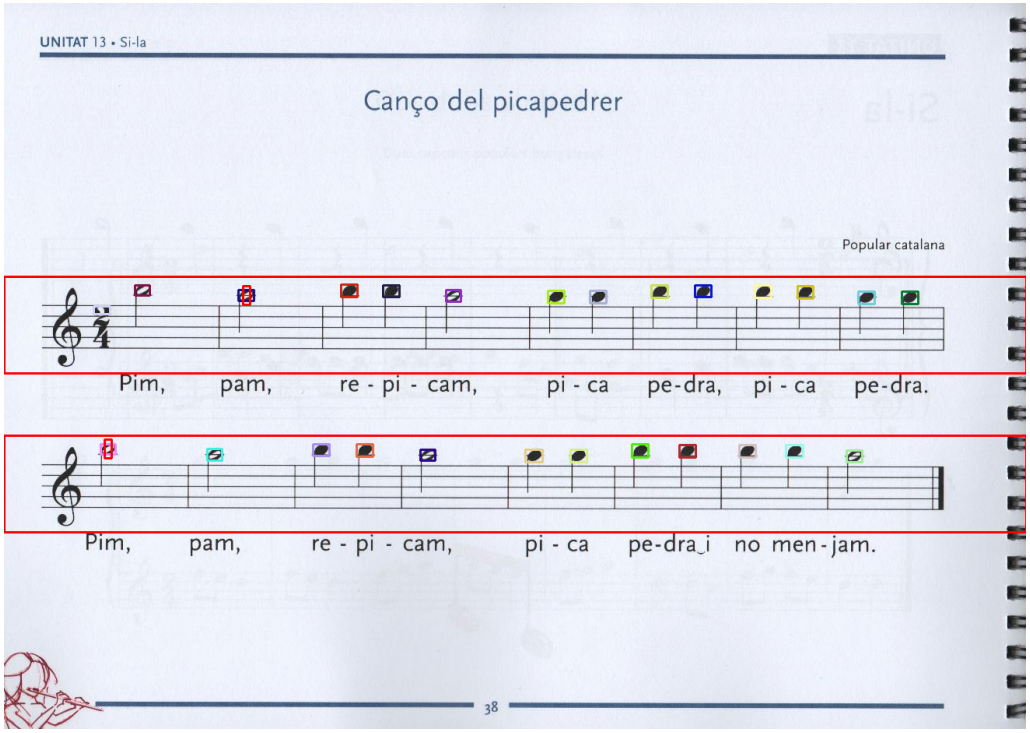


Figura D.86: Partitura amb els patrons reconeguts dibuixats **Font:** Pròpia

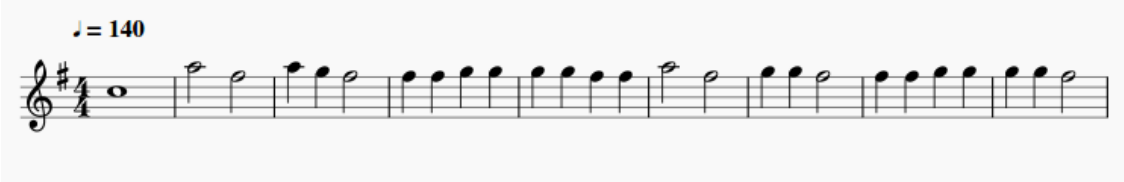


Figura D.87: Captura de pantalla de la partitura MIDI **Font:** Pròpia

Comentari: Partitura reconeguda al 48 %. El número 2 que indica el tipus de compàs inicial de la partitura l'ha reconegut com un Mi5 rodona. Al ser notes tan agudes no les ha reconegut bé. Les ha reconegut un interval d'una tercera major per sota.

Ara ve Nadal

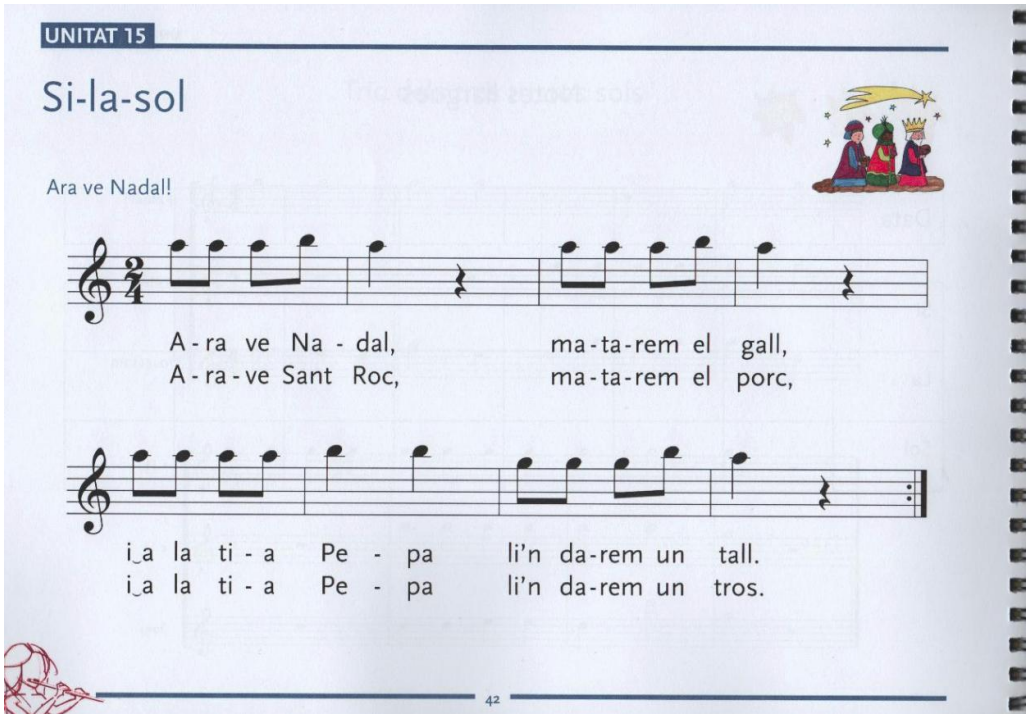


Figura D.88: Partitura Original
Font: Llibre "*Aprèn jugant amb la flauta travessera*", Volum 1

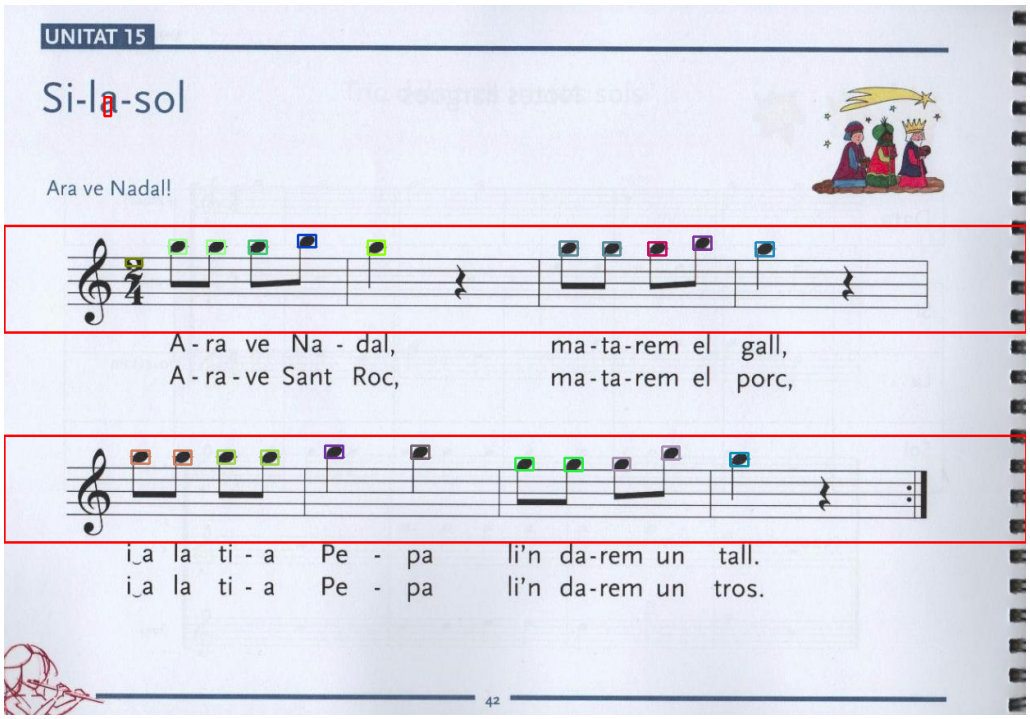


Figura D.89: Partitura amb els patrons reconeguts dibuixats **Font:** Pròpia

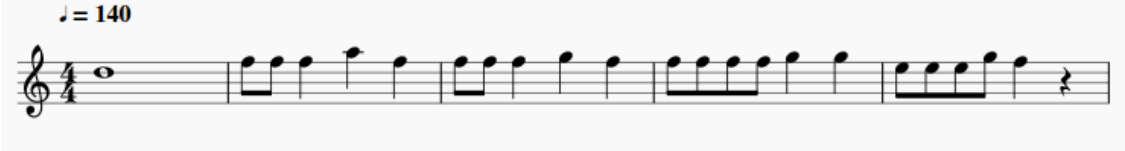


Figura D.90: Captura de pantalla de la partitura MIDI **Font:** Pròpia

Comentari: Partitura reconeguda al 38,1 %. El número 2 que indica el tipus de compàs inicial de la partitura l'ha reconegut com un Mi5 rodona. Al ser notes tan agudes no les ha reconegut bé. Les ha reconegut un interval d'una tercera major per sota. Ha reconegut un patró en el títol de la cançó però l'ha ignorat perquè està situat per sobre dels límits de les línies de pentagrama.

El Rellotge

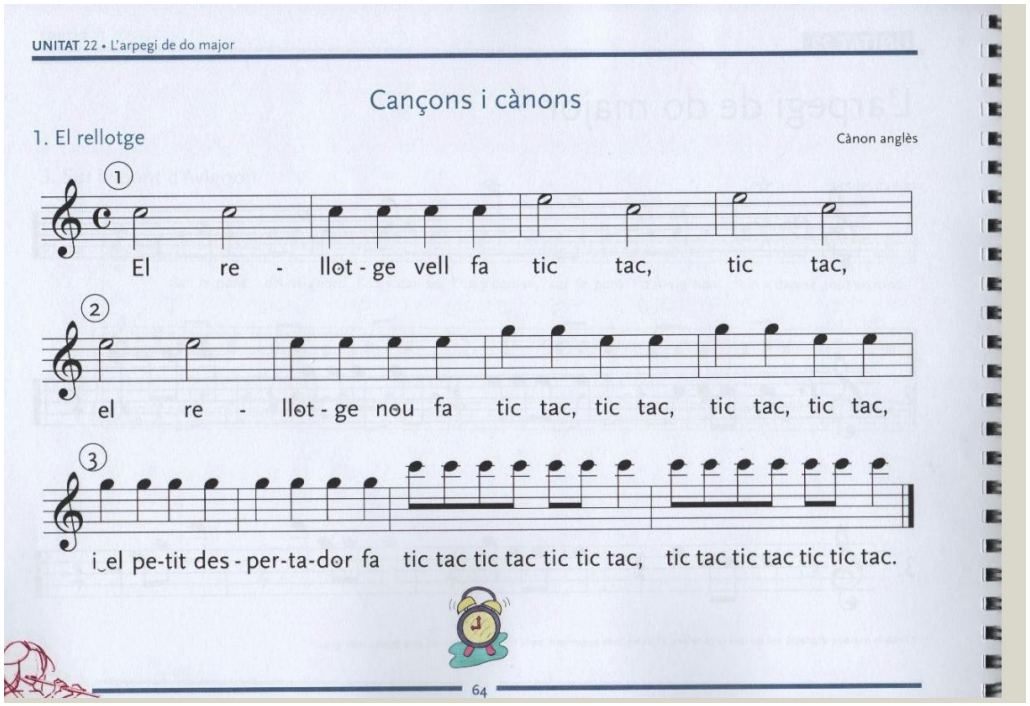


Figura D.91: Partitura Original
Font: Llibre "*Aprèn jugant amb la flauta travessera*", Volum 1

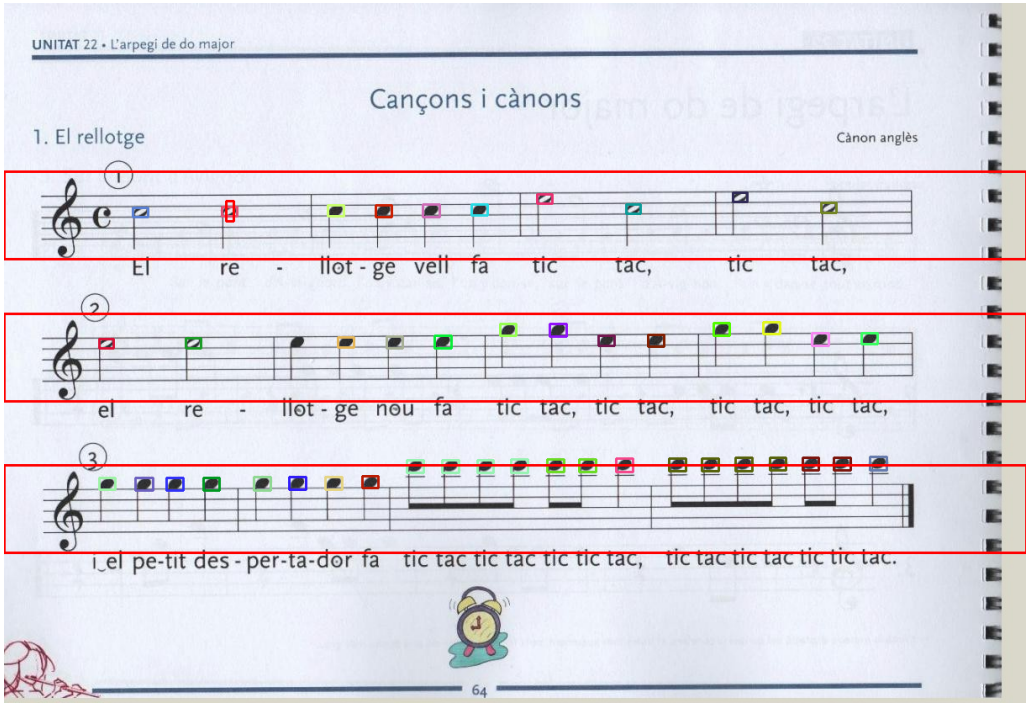


Figura D.92: Partitura amb els patrons reconeguts dibuixats **Font:** Pròpia

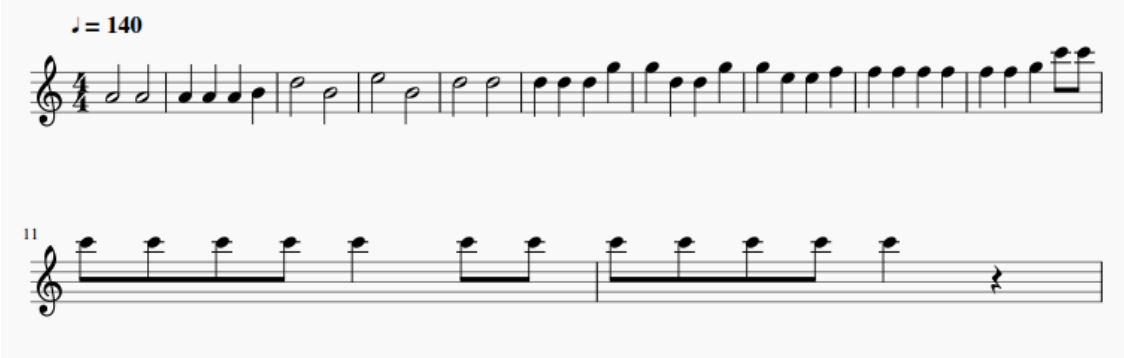


Figura D.93: Captura de pantalla de la partitura MIDI **Font:** Pròpia

Comentari: Partitura reconeguda al 64,13 %. El patró d'una negra no l'ha reconegut i hi ha una nota menys en total que a la imatge. Les notes no les ha reconegut bé fins les més agudes del final de la cançó.

APÈNDIX E

Codi del programa Music Scanner

A continuació mostro el codi dels 4 mòduls de Python.

E.1. Mòdul musicscanner.py

```
1  from files.reconeixement import *
2  from midiutil.MidiFile import MIDIFile
3  import os
4  from random import randint
5
6  pentagrama_fitxers = [
7      "files/patrons/pentagrama8.png",
8      ]
9  negra_fitxers = [
10     "files/patrons/negra.png",
11     "files/patrons/nota-negra.png"]
12 sostingut_fitxers = [
13     "files/patrons/f-sostingut.png",
14     "files/patrons/sostingut-linia.png",
15     "files/patrons/sostingut-espai.png",
16     "files/patrons/sostingut.png"]
17 bemoll_fitxers = [
18     "files/patrons/bemoll-linia.png",
19     "files/patrons/bemoll-espai.png" ]
20 blanca_fitxers = [
21     "files/patrons/blanca-espai.png",
22     "files/patrons/blanca-nota-linia.png",
```

```
23      "files/patrons/blanca-linia.png",
24      "files/patrons/blanca-nota-espai.png",
25      ]
26  rodona_fitxers = [
27      "files/patrons/rodona-espai.png",
28      "files/patrons/rodona-nota-linia.png",
29      "files/patrons/rodona-linia.png",
30      "files/patrons/rodona-nota-espai.png"]
31  silenci_negra_fitxers = [
32      "files/patrons/silenci-negra.png",
33      "files/patrons/silenci-negra-pent.png"]
34
35  BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__))
        ↪ )
36
37  pentagrama_imgs = [cv2.imread(pentagrama_fitxer, 0) for
        ↪ pentagrama_fitxer in pentagrama_fitxers]
38  negra_imgs = [cv2.imread(negra_fitxer, 0) for negra_fitxer in
        ↪ negra_fitxers]
39  sostingut_imgs = [cv2.imread(sostingut_fitxers, 0) for
        ↪ sostingut_fitxers in sostingut_fitxers]
40  bemoll_imgs = [cv2.imread(bemoll_fitxer, 0) for bemoll_fitxer in
        ↪ bemoll_fitxers]
41  blanca_imgs = [cv2.imread(blanca_fitxer, 0) for blanca_fitxer in
        ↪ blanca_fitxers]
42  rodona_imgs = [cv2.imread(rodona_fitxer, 0) for rodona_fitxer in
        ↪ rodona_fitxers]
```

```
43 rodona_imgs = [cv2.imread(rodona_fitxer, 0) for rodona_fitxer in
   ↪ rodona_fitxers]
44 silenci_negra_imgs = [cv2.imread(silenci_negra_fitxer, 0) for
   ↪ silenci_negra_fitxer in silenci_negra_fitxers]
45
46 pentagrama_inferior, pentagrama_superior, pentagrama_llindar = 70,
   ↪ 180, 0.70
47 sostingut_inferior, sostingut_superior, sostingut_llindar = 50, 80,
   ↪ 0.70
48 bemoll_inferior, bemoll_superior, bemoll_llindar = 40, 80, 0.77
49 negra_inferior, negra_superior, negra_llindar = 43, 80, 0.70
50 blanca_inferior, blanca_superior, blanca_llindar = 40, 80, 0.65
51 rodona_inferior, rodona_superior, rodona_llindar = 40, 80, 0.70
52 silenci_negra_inferior, silenci_negra_superior, silenci_negra_llindar
   ↪  = 65,80,0.70
53
54 def preprocessat_imatge(partitura):
55         img_color = cv2.imread(partitura, 1)
56         img = cv2.imread(partitura, 0)
57         h,w = img.shape[0],img.shape[1]
58         cv2.imwrite('img_color.png', img_color)
59         while(w > 1600):
60                 print("W␣%f,␣H␣%f",w,h)
61                 img_color = cv2.resize(img_color, None, fx = 0.9, fy =
                      ↪  0.9, interpolation = cv2.INTER_AREA)
62                 img = cv2.resize(img, None, fx = 0.9, fy = 0.9,
                      ↪ interpolation = cv2.INTER_AREA)
63                 h,w = img.shape[0],img.shape[1]
```

```
64          img_bw = img
65          ret3,img_bw = cv2.threshold(img_bw,127,255,cv2.THRESH_BINARY+
                ↪ cv2.THRESH_OTSU)
66          return img,img_bw,img_color,h,w
67
68  def scanner(partitura):
69          try:
70                  partitura = str(BASE_DIR) + "/media/documents/" + str(
                        ↪ partitura)
71                  img,img_bw,img_color,h,w = preprocessat_imatge(
                        ↪ partitura)
72
73                  rects_pentagrama = ubica_patro(img_bw, pentagrama_imgs
                        ↪ , pentagrama_inferior, pentagrama_superior,
                        ↪ pentagrama_llindar)
74                  rects_pentagrama = fusiona_rects([j for i in
                        ↪ rects_pentagrama for j in i], 0.01)
75                  rects_pentagrama_img = img_color.copy()
76                  for r in rects_pentagrama:
77                      r.dibuixa(rects_pentagrama_img, (0, 0, 255), 2)
78                  cv2.imwrite('rects_pentagrama_img.png',
                        ↪ rects_pentagrama_img)
79
80                  alt = rects_pentagrama[0].h
81                  linies_pentagrama = fusiona_rects([Rectangle(0, r.y, w
                        ↪ , r.h) for r in rects_pentagrama], 0.5)
82                  linies_pentagrama_img = img_color.copy()
83                  for r in linies_pentagrama:
```

```
84                  r.h = alt
85                  r.dibuixa(linies_pentagrama_img, (0, 0, 255), 2)
86              cv2.imwrite('rects_linies_pentagrama_img.png',
                    ↪ linies_pentagrama_img)
87
88              rects_sostingut = ubica_patro(img_bw, sostingut_imgs,
                    ↪ sostingut_inferior, sostingut_superior,
                    ↪ sostingut_llindar)
89              rects_sostingut = fusiona_rects([j for i in
                    ↪ rects_sostingut for j in i], 0.5)
90              rects_sostingut_img = img_color.copy()
91              for r in rects_sostingut:
92                  r.dibuixa(rects_sostingut_img, (0, 0, 255), 2)
93              cv2.imwrite('rects_sostingut_img.png',
                    ↪ rects_sostingut_img)
94
95              rects_bemoll = ubica_patro(img_bw, bemoll_imgs,
                    ↪ bemoll_inferior, bemoll_superior, bemoll_llindar
                    ↪ )
96              rects_bemoll = fusiona_rects([j for i in rects_bemoll
                    ↪ for j in i], 0.5)
97              rects_bemoll_img = img_color.copy()
98              for r in rects_bemoll:
99                  r.dibuixa(rects_bemoll_img, (0, 0, 255), 2)
100             cv2.imwrite('rects_bemoll_img.png', rects_bemoll_img)
101
102             rects_negres = ubica_patro(img_bw, negra_imgs,
                    ↪ negra_inferior, negra_superior, negra_llindar)
```

```
103             rects_negres = fusiona_rects([j for i in rects_negres
                    ↪ for j in i], 0.5)
104             rects_negres_img = img_color.copy()
105             for r in rects_negres:
106                 r.dibuixa(rects_negres_img, (0, 0, 255), 2)
107             cv2.imwrite('rects_negres_img.png', rects_negres_img)
108
109             rects_blanques = ubica_patro(img_bw, blanca_imgs,
                    ↪ blanca_inferior, blanca_superior, blanca_llindar
                    ↪ )
110             rects_blanques = fusiona_rects([j for i in
                    ↪ rects_blanques for j in i], 0.5)
111             rects_blanques_img = img_color.copy()
112             for r in rects_blanques:
113                 r.dibuixa(rects_blanques_img, (0, 0, 255), 2)
114             cv2.imwrite('rects_blanques_img.png',
                    ↪ rects_blanques_img)
115
116             rects_rodones = ubica_patro(img_bw, rodona_imgs,
                    ↪ rodona_inferior, rodona_superior, rodona_llindar
                    ↪ )
117             rects_rodones = fusiona_rects([j for i in
                    ↪ rects_rodones for j in i], 0.5)
118             rects_rodones_img = img_color.copy()
119             for r in rects_rodones:
120                 r.dibuixa(rects_rodones_img, (0, 0, 255), 2)
121             cv2.imwrite('rects_rodones_img.png', rects_rodones_img
                    ↪ )
```

```
122
123             rects_silenci_negra = ubica_patro(img_bw,
                    ↪ silenci_negra_imgs, silenci_negra_inferior,
                    ↪ silenci_negra_superior, silenci_negra_llindar)
124             rects_silenci_negra = fusiona_rects([j for i in
                    ↪ rects_silenci_negra for j in i], 0.5)
125             rects_silenci_negra_img = img_color.copy()
126             for r in rects_silenci_negra:
127                 r.dibuixa(rects_silenci_negra_img, (0, 0, 255), 2)
128             cv2.imwrite('rects_silenci_negra_img.png',
                    ↪ rects_silenci_negra_img)
129
130             elem_musicals = []
131             for linia in linies_pentagrama:
132                 sostinguts = [Nota(r, "sostingut", linia) for r in
                        ↪ rects_sostingut if abs(r.centre[1] - linia.
                        ↪ centre[1]) < linia.h]
133                 bemolls = [Nota(r, "bemoll", linia) for r in
                        ↪ rects_bemoll if abs(r.centre[1] - linia.
                        ↪ centre[1]) < linia.h]
134                 negres = [Nota(r, "4,8", linia, sostinguts, bemolls
                        ↪ )for r in rects_negres if abs(r.centre[1] -
                        ↪ linia.centre[1]) < linia.h]
135                 blanques = [Nota(r, "blanca", linia, sostinguts,
                        ↪ bemolls) for r in rects_blanques if abs(r.
                        ↪ centre[1] - linia.centre[1]) < linia.h]
136                 rodones = [Nota(r, "rodona", linia, sostinguts,
                        ↪ bemolls) for r in rects_rodones if abs(r.
```

```
                    ↪ centre[1] - linia.centre[1]) < linia.h]
137                 notes = negres + blanques + rodones
138                 notes.sort(key=lambda n: n.rect.x)
139
140                 pentagrames =[]
141                 for r in rects_pentagrama:
142                     if r.superposicio(linia) > 0:
143                         pentagrames.append(r)
144                 pentagrames.sort(key=lambda r: r.x)
145
146                 nota_color = (randint(0, 255), randint(0, 255),
                    ↪ randint(0, 255))
147                 elem_musical = []
148                 corxeres = []
149                 i = 0; j = 0;
150                 while(i < len(notes)):
151                     if (notes[i].rect.x > pentagrames[j].x and j <
                        ↪ len(pentagrames)):
152                         j += 1;
153                         if len(corxeres) > 0:
154                             elem_musicals.append(corxeres)
155                             corxeres = []
156                         nota_color = (randint(0, 255), randint(0,
                            ↪ 255), randint(0, 255))
157                     else:
158                         if len(pentagrames) < 15:
159                             break
160                         else:
```

```
161                             corxeres.append(notes[i])
162                             notes[i].rect.dibuixa(img_color,
                                    ↪ nota_color, 2)
163                             i += 1
164                     elem_musicals.append(corxeres)
165
166                 for r in linies_pentagrama:
167                     r.dibuixa(img_color, (0, 0, 255), 2)
168                 for r in rects_sostingut:
169                     r.dibuixa(img_color, (0, 0, 255), 2)
170                 rects_bemoll_img = img_color.copy()
171                 for r in rects_bemoll:
172                     r.dibuixa(img_color, (0, 0, 255), 2)
173                 for r in rects_silenci_negra:
174                     r.dibuixa(img_color, (0, 0, 255), 2)
175
176                 cv2.imwrite(partitura+'.png', img_color)
177
178                 for elem_musical in elem_musicals:
179                     print([ nota.nota + "␣" + nota.elem for nota in
                            ↪ elem_musical])
180
181                 midi = MIDIFile(1)
182                 track = 0
183                 time = 0
184                 channel = 0
185                 volume = 100
186                 midi.addTrackName(track, time, "Track")
```

```
187                midi.addTempo(track, time, 140)
188                for elem_musical in elem_musicals:
189                    durada = None
190                    for nota in elem_musical:
191                        tipus_nota = nota.elem
192                        if tipus_nota == "rodona":
193                            durada = 4
194                        elif tipus_nota == "blanca":
195                            durada = 2
196                        elif tipus_nota == "negra":
197                            if len(elem_musical) == 1:
198                                durada = 1
199                            else:
200                                durada = 0.5
201                        freq = nota.freq
202                        midi.addNote(track,channel,freq,time,durada,
                               ↪ volume)
203                        time += durada
204                arxiu_musica = open(partitura+'.mid', 'wb')
205                midi.writeFile(arxiu_musica)
206                arxiu_musica.close()
207                readmidi = os.path.join(BASE_DIR,"files","PySynth")
208                os.system("python3␣"+readmidi+"/readmidi.py␣"+
                       ↪ partitura+".mid␣1␣"+partitura+".wav")
209                return 1
210            except:
211                return 0
```

E.2. Mòdul reconeixement.py

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  import cv2
4  import os
5  import numpy as np
6  from files.rectangle import Rectangle
7  from files.nota import Nota
8
9  def deforma_patro(img, patrons, start_percent, stop_percent,llindar):
10     """
11     Els elements musicals poden variar en mida i tipus de lletra per
12     cada partitura en concret. Tenint en compte aquest fet, aquesta
13     funció varia la mida dels patrons per determinar si una plantilla
14     está present o no a la imatge a partir d'escalar els patrons i
15     determinar el millor patró en funció de les coincidẽncies entre
16     patró/imatge.
17     Els valors start/stop/llindar s'han ajustat a partir
18     d'experimentar amb diferents imatges i comprovar que són els
19     valors que encaixen més.
20     La funció retorna una llista amb les millors ubicacions i
21     l'escalat que causa els millors resultats (millors coinci-
22     dẽncies) amb la imatge.
23     """
24     millor_ubicacio_comptador = -1
25     millor_ubicacions = []
26     millor_escalat = 1
```

```
27      i = []
28      for e in range(start_percent, stop_percent + 1, 2):
29          i.append(e/100.0)
30      for escalat in (i):
31          ubicacions = []
32          ubicacio_comptador = 0
33          for patro in patrons:
34              patro = cv2.resize(patro, None, fx = escalat, fy = escalat
                    ↪ , interpolation = cv2.INTER_AREA)
35              result = cv2.matchTemplate(img, patro, cv2.
                    ↪ TM_CCOEFF_NORMED)
36              result = np.where(result >= llindar)
37              ubicacio_comptador += len(result[0])
38              ubicacions += [result]
39          print("escalat:␣{0},␣coincidẽncies:␣{1}".format(escalat,
                ↪ ubicacio_comptador))
40          if (ubicacio_comptador > millor_ubicacio_comptador):
41              millor_ubicacio_comptador = ubicacio_comptador
42              millor_ubicacions = ubicacions
43              millor_escalat = escalat
44          elif (ubicacio_comptador < millor_ubicacio_comptador):
45              pass
46      return millor_ubicacions, millor_escalat
47
48  def ubica_patro(img, patrons, start, stop, llindar):
49      """
50      A partir de la funció deforma_patro, guarda els objectes
            ↪ rectangle corresponents als patrons i en retorna la llista
```

```
        ↪ de rectangles.
51      """
52      ubicacions, escalat = deforma_patro(img, patrons, start, stop,
            ↪ llindar)
53      img_ubicacions = []
54      for i in range(len(patrons)):
55          w, h = patrons[i].shape[::-1]
56          w *= escalat
57          h *= escalat
58          print("Mida␣rectangles␣w,h␣=",w,h)
59          img_ubicacions.append([Rectangle(pt[0], pt[1], w, h) for pt
                ↪ in zip(*ubicacions[i][::-1])])
60      return img_ubicacions
61
62  def fusiona_rects(rects, llindar):
63      """
64      A partir de la llista de rectangles , rects i un llindar, fusiona
            ↪  els rectangles que se sobreposen i els pinta només
            ↪ remarcats pel contorn i de mida llindar
65      """
66      rects_filtrats = []
67      while len(rects) > 0:
68          r = rects.pop(0)
69          rects.sort(key=lambda rec: rec.distancia(r))
70          fusionat = True
71          while(fusionat):
72              fusionat = False
73              i = 0
```

```
74              for _ in range(len(rects)):
75                  if r.superposicio(rects[i]) > llindar or rects[i].
                        ↪ superposicio(r) > llindar:
76                      r = r.fusiona(rects.pop(i))
77                      fusionat = True
78                  elif rects[i].distancia(r) > r.w/2 + rects[i].w/2:
79                      break
80                  else:
81                      i += 1
82          rects_filtrats.append(r)
83      return rects_filtrats
```

E.3. Mòdul Rectangle.py

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4  import cv2
5  import math
6
7  class Rectangle(object):
8      def __init__(self, x, y, w, h):
9          """
10         Inicialitza l'objecte Rectangle amb la posició (x,y), amplada
               ↪ i altura
11         També calcula el punt mig i l'área
12         """
13         self.x = x;
```

```
14          self.y = y;
15          self.w = w;
16          self.h = h;
17          self.centre = self.x + self.w/2.0, self.y + self.h/2.0
18          self.area = self.w * self.h
19
20      def __str__(self):
21          return "Rectangle:␣x=%s,␣y=%s,␣w=%s,␣h=%s" % (self.x,self.y,
                ↪ self.w,self.h)
22
23      def superposicio(self, other):
24          """
25          Retorna si self está dins del rectangle other, o <1 en funció
                ↪  del tros de self que está dins del rectangle other.
26          >>> r = Rectangle(1,1,3,2)
27          >>> s = Rectangle(1,1,4,2)
28          >>> r.superposicio(s)
29          1
30          >>> s.superposicio(r)
31          0.75
32          """
33          superposicio_x = max(0, min(self.x + self.w, other.x + other.
                ↪ w) - max(self.x, other.x));
34          superposicio_y = max(0, min(self.y + self.h, other.y + other.
                ↪ h) - max(self.y, other.y));
35          superposicio_area = superposicio_x * superposicio_y
36          return superposicio_area / float(self.area)
37
```

```
38      def distancia(self, other):
39          """
40          Retorna la distáncia entre els dos centres dels rectangles
41          >>> r = Rectangle(1,1,3,2)
42          >>> s = Rectangle(1,1,4,2)
43          >>> r.distancia(s)
44          0.5
45          >>> e = Rectangle(1,1,4,2)
46          >>> f = Rectangle(4,2,2,3)
47          >>> e.distancia(f)
48          2.5
49          """
50          dx = self.centre[0] - other.centre[0]
51          dy = self.centre[1] - other.centre[1]
52          return math.sqrt(dx*dx + dy*dy)
53
54      def fusiona(self, other):
55          """
56          A partir de dos rectangles, et retorna un objecte rectangle
              ↪ que és la unió dels dos inicials
57          >>> r = Rectangle(1,1,3,2)
58          >>> s = Rectangle(1,1,4,2)
59          >>> print str(r.fusiona(s))
60          Rectangle: x=1, y=1, w=4, h=2
61          >>> e = Rectangle(1,1,4,2)
62          >>> f = Rectangle(4,2,2,3)
63          >>> print str(e.fusiona(f))
64          Rectangle: x=1, y=1, w=5, h=4
```

```
65          """
66          x = min(self.x, other.x)
67          y = min(self.y, other.y)
68          w = max(self.x + self.w, other.x + other.w) - x
69          h = max(self.y + self.h, other.y + other.h) - y
70          return Rectangle(x, y, w, h)
71
72      def dibuixa(self, img, color, thickness):
73          """
74          Dibuixa un rectangle a la imatge img a les coordenades
              ↪ donades, començant des de la cantonada superior
              ↪ esquerra (0,0)
75          """
76          pos = ((int)(self.x), (int)(self.y))
77          size = ((int)(self.x + self.w), (int)(self.y + self.h))
78          cv2.rectangle(img, pos, size, color, thickness)
```

E.4. Mòdul Nota.py

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4  from files.rectangle import Rectangle
5
6  nota_defs = {
7       0 : ("c6", 84),
8       1 : ("b5", 83),
9       2 : ("a5", 81),
```

```
10      3 : ("g5", 79),
11      4 : ("f5", 77),
12      5 : ("e5", 76),
13      6 : ("d5", 74),
14      7 : ("c5", 72),
15      8 : ("b4", 71),
16      9 : ("a4", 69),
17      10: ("g4", 67),
18      11: ("f4", 65),
19      12: ("e4", 64),
20      13: ("d4", 62),
21      14: ("c4", 60),
22      15: ("b3", 59),
23      16: ("a3", 57),
24      17: ("g3", 55),
25      18: ("f3", 53),
26      19: ("e3", 52),
27      20: ("d3", 50),
28      21: ("c3", 48),
29      22: ("b2", 47),
30      23: ("a2", 45),
31      24: ("f2", 41),
32      }
33
34  class Nota(object):
35    """
36    Crea un objecte Nota independentment de si és una nota (rodona,
        ↪ blanca,negra),
```

```
37  sostingut o bemoll.
38  El que ens interessa més de l'objecte Nota és nota_def, que ens diu
        ↪ la altura
39  d'aquest objecte (do,re,mi,fa,sol,la,si) en qualsevol octava
40  Per les alteracions, voldrem comparar la nota (do,re...si) sense
        ↪ importar la
41  alćada (do4,do5,...) ja que una alteració ho és a totes les notes
        ↪ independent-
42  ment de l'alćada
43  """
44  def __init__(self, rect, elem, linia_rect, sostinguts = [], bemolls
        ↪ = []):
45    self.rect = rect
46    self.elem = elem
47    self.linia_rect = linia_rect
48    self.sostinguts = sostinguts
49    self.bemolls = bemolls
50    nota_real = rect.centre[1] - linia_rect.y
51
52    if round(17*linia_rect.h/87.6) <17:
53      total_notes = 17.5
54    elif round(17*linia_rect.h/87.6) >=21:
55      total_notes = 20.5
56    else:
57      total_notes = round(17*linia_rect.h/87.6) + 0.5
58    step = linia_rect.h/total_notes
59
60    i = 1
```

```
61    for i in range(len(nota_defs)):
62      if (nota_real >= (i*step - step) and nota_real <= (i*step +
          ↪ step)):
63        nota_def = nota_defs[i]
64        break
65      else:
66        i+=1
67
68    self.nota = nota_def[0]
69    self.freq = nota_def[1]
70
71    ordre_sost = ['f','c','g','d','a','e','b']
72    i = 0
73    sost = ordre_sost[0:len(sostinguts)]
74    if len(sost) > 0:
75      for e in sost:
76        if self.nota[0] == e:
77          self.nota += "#"
78          self.freq += 1
79    ordre_bem = ['b','e','a','d','g','c','f']
80    i = 0
81    bem = ordre_bem[0:len(bemolls)]
82    if len(bem) > 0:
83      for e in bem:
84        if self.nota[0] == e:
85          self.nota += "b"
86          self.freq -= 1
```

APÈNDIX F

Resultats inspecció seguretat del servidor web

A continuació mostro els resultats de la inspecció de seguretat de la pàgina web joanbruch.pythonanywhere.com realitzada per SSL Labs [40].

SSL Report: joanbruch.pythonanywhere.com (35.173.69.207)

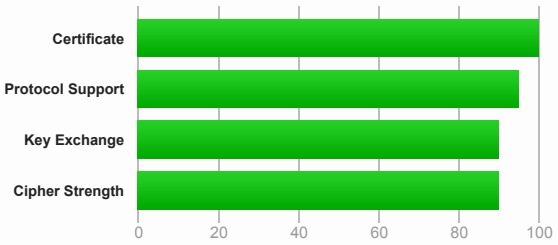
Assessed on: Thu, 24 May 2018 16:36:42 UTC | Hide | Clear cache

Scan Another »

Summary

Overall Rating

A



Visit our documentation page for more information, configuration guides, and books. Known issues are documented here.

DNS Certification Authority Authorization (CAA) Policy found for this domain. MORE INFO »

Certificate #1: RSA 2048 bits (SHA256withRSA)

Server Key and Certificate #1

Subject	*.pythonanywhere.com Fingerprint SHA256: 83628246001d1f4fdb6d79359e7c1e82058bbe33a89b2517eb0fc1f16adb353e Pin SHA256: Brq6SHEBk0tvFewSUZiT1Evbn9SiaAAJ7F2DoMkrXq4=
Common names	*.pythonanywhere.com
Alternative names	*.pythonanywhere.com pythonanywhere.com
Serial Number	4fbe2d0931f3924a37de35dc4253328f
Valid from	Mon, 25 Sep 2017 00:00:00 UTC
Valid until	Sun, 09 Dec 2018 23:59:59 UTC (expires in 6 months and 15 days)
Key	RSA 2048 bits (e 65537)
Weak key (Debian)	No
Issuer	COMODO RSA Domain Validation Secure Server CA AIA: http://crt.comodoca.com/COMODORSADomainValidationSecureServerCA.crt
Signature algorithm	SHA256withRSA
Extended Validation	No
Certificate Transparency	No
OCSP Must Staple	No
Revocation information	CRL, OCSP CRL: http://crl.comodoca.com/COMODORSADomainValidationSecureServerCA.crl OCSP: http://ocsp.comodoca.com
Revocation status	Good (not revoked)
DNS CAA	Yes policy host: pythonanywhere.com issue: comodoca.com flags:0 issuewild: comodoca.com flags:0
Trusted	Yes Mozilla Apple Android Java Windows

Additional Certificates (if supplied)

Certificates provided	4 (5412 bytes)
Chain issues	Contains anchor

#2

Additional Certificates (if supplied)

Subject	COMODO RSA Domain Validation Secure Server CA Fingerprint SHA256: 02ab57e4e67a0cb48dd2ff34830e8ac40f4476fb08ca6be3f5cd846f646840f0 Pin SHA256: klO23nT2ehFDXCfx3eHTDRESMz3asj1muO+4aIdjiuY=
Valid until	Sun, 11 Feb 2029 23:59:59 UTC (expires in 10 years and 8 months)
Key	RSA 2048 bits (e 65537)
Issuer	COMODO RSA Certification Authority
Signature algorithm	SHA384withRSA

#3

Subject	COMODO RSA Certification Authority Fingerprint SHA256: 4f32d5dc00f715250abcc486511e37f501a899deb3bf7ea8adbbd3aef1c412da Pin SHA256: grX4Ta9HpZx6tSHkmCrvpApTQGo67CYDnvprLg5yRME=
Valid until	Sat, 30 May 2020 10:48:38 UTC (expires in 2 years)
Key	RSA 4096 bits (e 65537)
Issuer	AddTrust External CA Root
Signature algorithm	SHA384withRSA

#4

Subject	AddTrust External CA Root In trust store Fingerprint SHA256: 687fa451382278fff0c8b11f8d43d576671c6eb2bceab413fb83d965d06d2ff2 Pin SHA256: ICppFqbkrlJ3EcVFAkeip0+44VaoJUymbnOaEUk7tEU=
Valid until	Sat, 30 May 2020 10:48:38 UTC (expires in 2 years)
Key	RSA 2048 bits (e 65537)
Issuer	AddTrust External CA Root Self-signed
Signature algorithm	SHA1withRSA Weak, but no impact on root certificate

Certification Paths

Mozilla | Apple | **Android** | Java | Windows

Path #1: Trusted

1	Sent by server	*.pythonanywhere.com Fingerprint SHA256: 83628246001d1f4fdb6d79359e7c1e82058bbe33a89b2517eb0fc1f16adb353e Pin SHA256: Brq6SHEBk0tvFewSUZiT1Evbn9SiaAAJ7F2DoMkrXq4= RSA 2048 bits (e 65537) / SHA256withRSA
2	Sent by server	COMODO RSA Domain Validation Secure Server CA Fingerprint SHA256: 02ab57e4e67a0cb48dd2ff34830e8ac40f4476fb08ca6be3f5cd846f646840f0 Pin SHA256: klO23nT2ehFDXCfx3eHTDRESMz3asj1muO+4aIdjiuY= RSA 2048 bits (e 65537) / SHA384withRSA
3	In trust store	COMODO RSA Certification Authority Self-signed Fingerprint SHA256: 52f0e1c4e58ec629291b60317f074671b85d7ea80d5b07273463534b32b40234 Pin SHA256: grX4Ta9HpZx6tSHkmCrvpApTQGo67CYDnvprLg5yRME= RSA 4096 bits (e 65537) / SHA384withRSA

Path #2: Trusted

1	Sent by server	*.pythonanywhere.com Fingerprint SHA256: 83628246001d1f4fdb6d79359e7c1e82058bbe33a89b2517eb0fc1f16adb353e Pin SHA256: Brq6SHEBk0tvFewSUZiT1Evbn9SiaAAJ7F2DoMkrXq4= RSA 2048 bits (e 65537) / SHA256withRSA
2	Sent by server	COMODO RSA Domain Validation Secure Server CA Fingerprint SHA256: 02ab57e4e67a0cb48dd2ff34830e8ac40f4476fb08ca6be3f5cd846f646840f0 Pin SHA256: klO23nT2ehFDXCfx3eHTDRESMz3asj1muO+4aIdjiuY= RSA 2048 bits (e 65537) / SHA384withRSA
3	Sent by server	COMODO RSA Certification Authority Fingerprint SHA256: 4f32d5dc00f715250abcc486511e37f501a899deb3bf7ea8adbbd3aef1c412da Pin SHA256: grX4Ta9HpZx6tSHkmCrvpApTQGo67CYDnvprLg5yRME= RSA 4096 bits (e 65537) / SHA384withRSA
4	Sent by server In trust store	AddTrust External CA Root Self-signed Fingerprint SHA256: 687fa451382278fff0c8b11f8d43d576671c6eb2bceab413fb83d965d06d2ff2 Pin SHA256: ICppFqbkrlJ3EcVFAkeip0+44VaoJUymbnOaEUk7tEU= RSA 2048 bits (e 65537) / SHA1withRSA Weak or insecure signature, but no impact on root certificate

Configuration

Protocols

TLS 1.3	No
TLS 1.2	Yes
TLS 1.1	Yes
TLS 1.0	Yes
SSL 3	No
SSL 2	No

For TLS 1.3 tests, we currently support draft version 18.

Cipher Suites

TLS 1.2 (suites in server-preferred order)

TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)	ECDH secp256r1 (eq. 3072 bits RSA) FS	256
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)	ECDH secp256r1 (eq. 3072 bits RSA) FS	128
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (0x9f)	DH 2048 bits FS	256
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 (0x9e)	DH 2048 bits FS	128
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xc028)	ECDH secp256r1 (eq. 3072 bits RSA) FS	256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)	ECDH secp256r1 (eq. 3072 bits RSA) FS	256
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 (0x6b)	DH 2048 bits FS	256
TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x39)	DH 2048 bits FS	256
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (0xc027)	ECDH secp256r1 (eq. 3072 bits RSA) FS	128
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)	ECDH secp256r1 (eq. 3072 bits RSA) FS	128
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 (0x67)	DH 2048 bits FS	128
TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x33)	DH 2048 bits FS	128
TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA (0xc012)	ECDH secp256r1 (eq. 3072 bits RSA) FS **WEAK**	112
TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA (0x16)	DH 2048 bits FS **WEAK**	112
TLS_RSA_WITH_AES_256_GCM_SHA384 (0x9d)	**WEAK**	256
TLS_RSA_WITH_AES_128_GCM_SHA256 (0x9c)	**WEAK**	128
TLS_RSA_WITH_AES_256_CBC_SHA256 (0x3d)	**WEAK**	256
TLS_RSA_WITH_AES_128_CBC_SHA256 (0x3c)	**WEAK**	128
TLS_RSA_WITH_AES_256_CBC_SHA (0x35)	**WEAK**	256
TLS_RSA_WITH_AES_128_CBC_SHA (0x2f)	**WEAK**	128
TLS_RSA_WITH_3DES_EDE_CBC_SHA (0xa)	**WEAK**	112
TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA (0x88)	DH 2048 bits FS	256
TLS_RSA_WITH_CAMELLIA_256_CBC_SHA (0x84)	**WEAK**	256
TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA (0x45)	DH 2048 bits FS	128
TLS_RSA_WITH_CAMELLIA_128_CBC_SHA (0x41)	**WEAK**	128

TLS 1.1 (suites in server-preferred order)

TLS 1.0 (suites in server-preferred order)

Handshake Simulation

Android 2.3.7 No SNI [2]	RSA 2048 (SHA256)	TLS 1.0	TLS_DHE_RSA_WITH_AES_128_CBC_SHA DH 2048 FS
Android 4.0.4	RSA 2048 (SHA256)	TLS 1.0	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA ECDH secp256r1 FS
Android 4.1.1	RSA 2048 (SHA256)	TLS 1.0	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA ECDH secp256r1 FS
Android 4.2.2	RSA 2048 (SHA256)	TLS 1.0	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA ECDH secp256r1 FS
Android 4.3	RSA 2048 (SHA256)	TLS 1.0	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA ECDH secp256r1 FS
Android 4.4.2	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 ECDH secp256r1 FS
Android 5.0.0	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 ECDH secp256r1 FS
Android 6.0	RSA 2048 (SHA256)	TLS 1.2 > http/1.1	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 ECDH secp256r1 FS
Android 7.0	RSA 2048 (SHA256)	TLS 1.2 > http/1.1	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 ECDH secp256r1 FS
Baidu Jan 2015	RSA 2048 (SHA256)	TLS 1.0	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA ECDH secp256r1 FS
BingPreview Jan 2015	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 ECDH secp256r1 FS
Chrome 49 / XP SP3	RSA 2048 (SHA256)	TLS 1.2 > http/1.1	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 ECDH secp256r1 FS
Chrome 57 / Win 7 R	RSA 2048 (SHA256)	TLS 1.2 > http/1.1	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 ECDH secp256r1 FS

Handshake Simulation

Firefox 31.3.0 ESR / Win 7	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 ECDH secp256r1 FS
Firefox 47 / Win 7 R	RSA 2048 (SHA256)	TLS 1.2 > http/1.1	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 ECDH secp256r1 FS
Firefox 49 / XP SP3	RSA 2048 (SHA256)	TLS 1.2 > http/1.1	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 ECDH secp256r1 FS
Firefox 53 / Win 7 R	RSA 2048 (SHA256)	TLS 1.2 > http/1.1	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 ECDH secp256r1 FS
Googlebot Feb 2018	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 ECDH secp256r1 FS
IE 7 / Vista	RSA 2048 (SHA256)	TLS 1.0	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA ECDH secp256r1 FS
IE 8 / XP No FS [1] No SNI [2]	RSA 2048 (SHA256)	TLS 1.0	TLS_RSA_WITH_3DES_EDE_CBC_SHA
IE 8-10 / Win 7 R	RSA 2048 (SHA256)	TLS 1.0	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA ECDH secp256r1 FS
IE 11 / Win 7 R	RSA 2048 (SHA256)	TLS 1.2	TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 DH 2048 FS
IE 11 / Win 8.1 R	RSA 2048 (SHA256)	TLS 1.2 > http/1.1	TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 DH 2048 FS
IE 10 / Win Phone 8.0	RSA 2048 (SHA256)	TLS 1.0	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA ECDH secp256r1 FS
IE 11 / Win Phone 8.1 R	RSA 2048 (SHA256)	TLS 1.2 > http/1.1	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA ECDH secp256r1 FS
IE 11 / Win Phone 8.1 Update R	RSA 2048 (SHA256)	TLS 1.2 > http/1.1	TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 DH 2048 FS
IE 11 / Win 10 R	RSA 2048 (SHA256)	TLS 1.2 > http/1.1	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 ECDH secp256r1 FS
Edge 15 / Win 10 R	RSA 2048 (SHA256)	TLS 1.2 > http/1.1	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 ECDH secp256r1 FS
Edge 13 / Win Phone 10 R	RSA 2048 (SHA256)	TLS 1.2 > http/1.1	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 ECDH secp256r1 FS
Java 6u45 No SNI [2]	Client does not support DH parameters > 1024 bits RSA 2048 (SHA256) | TLS 1.0 | TLS_DHE_RSA_WITH_AES_128_CBC_SHA | DH 2048		
Java 7u25	RSA 2048 (SHA256)	TLS 1.0	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA ECDH secp256r1 FS
Java 8u161	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 ECDH secp256r1 FS
OpenSSL 0.9.8y	RSA 2048 (SHA256)	TLS 1.0	TLS_DHE_RSA_WITH_AES_256_CBC_SHA DH 2048 FS
OpenSSL 1.0.1l R	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 ECDH secp256r1 FS
OpenSSL 1.0.2e R	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 ECDH secp256r1 FS
Safari 5.1.9 / OS X 10.6.8	RSA 2048 (SHA256)	TLS 1.0	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA ECDH secp256r1 FS
Safari 6 / iOS 6.0.1	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 ECDH secp256r1 FS
Safari 6.0.4 / OS X 10.8.4 R	RSA 2048 (SHA256)	TLS 1.0	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA ECDH secp256r1 FS
Safari 7 / iOS 7.1 R	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 ECDH secp256r1 FS
Safari 7 / OS X 10.9 R	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 ECDH secp256r1 FS
Safari 8 / iOS 8.4 R	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 ECDH secp256r1 FS
Safari 8 / OS X 10.10 R	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 ECDH secp256r1 FS
Safari 9 / iOS 9 R	RSA 2048 (SHA256)	TLS 1.2 > http/1.1	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 ECDH secp256r1 FS
Safari 9 / OS X 10.11 R	RSA 2048 (SHA256)	TLS 1.2 > http/1.1	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 ECDH secp256r1 FS
Safari 10 / iOS 10 R	RSA 2048 (SHA256)	TLS 1.2 > http/1.1	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 ECDH secp256r1 FS
Safari 10 / OS X 10.12 R	RSA 2048 (SHA256)	TLS 1.2 > http/1.1	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 ECDH secp256r1 FS
Apple ATS 9 / iOS 9 R	RSA 2048 (SHA256)	TLS 1.2 > http/1.1	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 ECDH secp256r1 FS
Yahoo Slurp Jan 2015	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 ECDH secp256r1 FS
YandexBot Jan 2015	RSA 2048 (SHA256)	TLS 1.2	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 ECDH secp256r1 FS

Not simulated clients (Protocol mismatch)

IE 6 / XP No FS [1] No SNI [2]	Protocol mismatch (not simulated)

(1) Clients that do not support Forward Secrecy (FS) are excluded when determining support for it.

(2) No support for virtual SSL hosting (SNI). Connects to the default site if the server uses SNI.

(3) Only first connection attempt simulated. Browsers sometimes retry with a lower protocol version.

(R) Denotes a reference browser or client, with which we expect better effective security.

(All) We use defaults, but some platforms do not use their best protocols and features (e.g., Java 6 & 7, older IE).

(All) Certificate trust is not checked in handshake simulation, we only perform TLS handshake.

Protocol Details

DROWN	No, server keys and hostname not seen elsewhere with SSLv2 **(1) For a better understanding of this test, please read this longer explanation** (2) Key usage data kindly provided by the Censys network search engine; original DROWN website here (3) Censys data is only indicative of possible key and certificate reuse; possibly out-of-date and not complete
Secure Renegotiation	**Supported**
Secure Client-Initiated Renegotiation	No
Insecure Client-Initiated Renegotiation	No
BEAST attack	Not mitigated server-side (more info) TLS 1.0: 0xc014
POODLE (SSLv3)	No, SSL 3 not supported (more info)

Protocol Details

POODLE (TLS)	No (more info)
Downgrade attack prevention	**Yes, TLS_FALLBACK_SCSV supported** (more info)
SSL/TLS compression	No
RC4	No
Heartbeat (extension)	Yes
Heartbleed (vulnerability)	No (more info)
Ticketbleed (vulnerability)	No (more info)
OpenSSL CCS vuln. (CVE-2014-0224)	No (more info)
OpenSSL Padding Oracle vuln. (CVE-2016-2107)	No (more info)
ROBOT (vulnerability)	No (more info)
Forward Secrecy	**Yes (with most browsers) ROBUST** (more info)
ALPN	Yes http/1.1
NPN	Yes http/1.1
Session resumption (caching)	Yes
Session resumption (tickets)	Yes
OCSP stapling	No
Strict Transport Security (HSTS)	No
HSTS Preloading	Not in: Chrome Edge Firefox IE
Public Key Pinning (HPKP)	No (more info)
Public Key Pinning Report-Only	No
Public Key Pinning (Static)	No (more info)
Long handshake intolerance	No
TLS extension intolerance	No
TLS version intolerance	No
Incorrect SNI alerts	No
Uses common DH primes	No
DH public server param (Ys) reuse	No
ECDH public server param reuse	No
Supported Named Groups	secp256r1
SSL 2 handshake compatibility	Yes

HTTP Requests

1 **https://joanbruch.pythonanywhere.com/** (HTTP/1.1 200 OK)

Miscellaneous

Test date	Thu, 24 May 2018 16:34:28 UTC
Test duration	134.552 seconds
HTTP status code	200
HTTP server signature	openresty/1.9.15.1
Server hostname	ec2-35-173-69-207.compute-1.amazonaws.com

SSL Report v1.31.0